

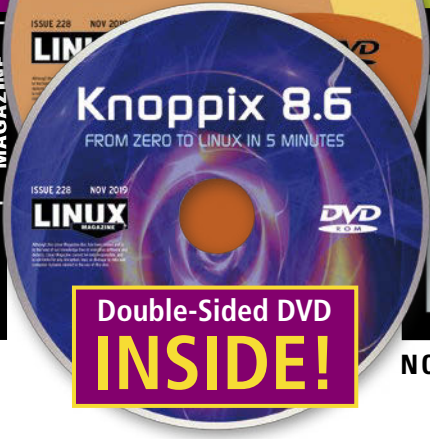
FREE DVD

**Super Grub2**  
Get back to your GNU/Linux & Windows computers!

**TASMOTA**  
Put new firmware on IoT devices

**SECRET FILE SHARING**  
with **OnionShare**

**LINUX**  
MAGAZINE



Double-Sided DVD  
**INSIDE!**

# LINUX

PRO

## MAGAZINE

NOVEMBER 2019

# ANONYMOUS FILE SHARING

Pass files secretly with OnionShare

**Baobab: Analyze disk usage**

**Sayonara Music Player**

Mingle your music collection with podcasts and Internet radio

**Safe Login**

Tools for better online password protection



**HiFiBerry**

Use a Rasp Pi to resurrect your old CD player

**/usr Merge**

Debian restructures the all-important /usr directory

**Love for LoRa**

Wireless networking at a distance

# LINUXVOICE

- Lutris open gaming platform
- McFly: Add AI to Bash
- maddog: Jump-start your business with FOSSH



**FOSSPicks**

- Gaia Sky
- Firejail app sandbox
- LIKO-12 retro-gaming engine

**Tutorial**

- Microblogging with Mastodon

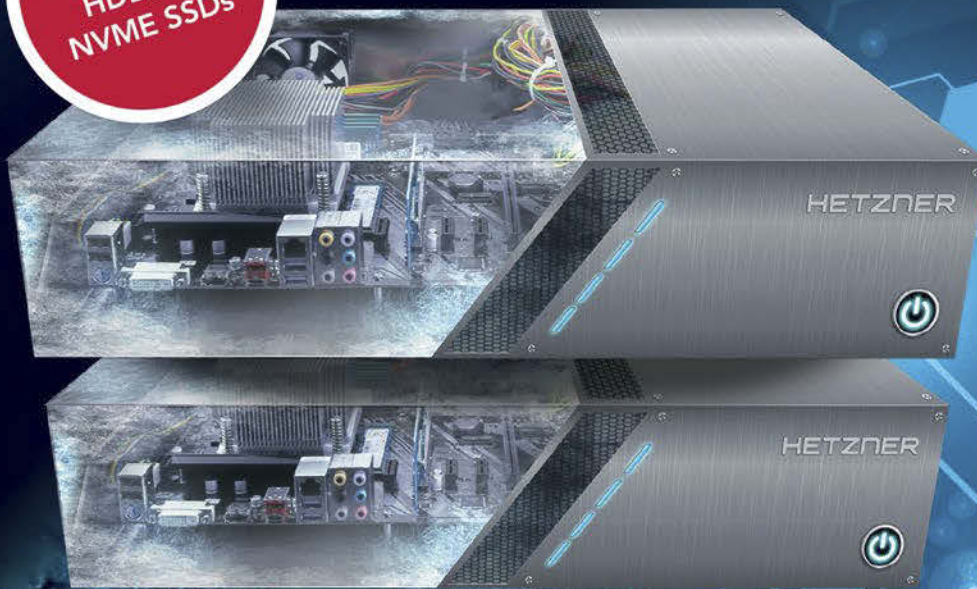
Issue 228  
Nov 2019  
US\$ 15.99  
CAN\$ 18.50

# HETZNER

# DEDICATED ROOT SERVER EX62

High speed performance with the new  
Intel Core i9-9900K octa-core processor

ENTERPRISE  
HDDs or  
NVME SSDs



## Dedicated Root Server EX62

- ✓ Intel® Core™ i9-9900K Octa-Core incl. Hyper-Threading-Technology
- ✓ 64 GB DDR4 RAM
- ✓ 2 x 8 TB SATA Enterprise Hard Drive 7200 rpm
- ✓ 100 GB Backup Space
- ✓ Traffic unlimited
- ✓ Location Germany or Finland
- ✓ No minimum contract
- ✓ Setup Fee \$78

monthly from \$ **72.50**

## Dedicated Root Server EX62-NVMe

- ✓ Intel® Core™ i9-9900 Octa-Core incl. Hyper-Threading-Technology
- ✓ 64 GB DDR4 RAM
- ✓ 2 x 1 TB NVMe SSD
- ✓ 100 GB Backup Space
- ✓ Traffic unlimited
- ✓ Location Germany or Finland
- ✓ No minimum contract
- ✓ Setup Fee \$78

monthly from \$ **72.50**

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.

[www.hetzner.com](http://www.hetzner.com)



# WON'T HAPPEN

Dear Reader,

In the latest update in the ongoing rapprochement between Microsoft and the FOSS community, several news sources report that Free Software legend Richard Stallman actually arrived at Microsoft Research headquarters this month and gave a talk for Microsoft employees [1]. As I have often used this space to mention in the past, this spirit of cooperation between Microsoft and the FOSS community is so unexpected that it actually seems a little unreal to those of us who remember the wild rhetoric and unrestrained legal wrangling of the bad old days.

As amazing as it might seem that Microsoft invited Richard Stallman to speak, people seemed more amazed that Richard Stallman actually *accepted* the invitation. Back in the days when Microsoft had a lot to say about Free Software, Stallman certainly had a lot to say about Microsoft. To this day, Stallman has a page on his personal website entitled “Reasons Not to Use Microsoft” with a fiery list of Redmond transgressions [2].

“How could this happen?” the pundits are asking. Richard Stallman is a firebrand, a provocateur. What is he doing at Microsoft? But that’s where it gets interesting. Richard Stallman is an very intelligent fellow, and he didn’t build the Free Software movement to where it is now through stony silence in the face of opposition. He built it by talking to people. Through the years, he has been uncompromising in his principles, but he has never been shy about presenting his case. Another thing to know is that Stallman doesn’t fall neatly into the over-simplified Windows versus Linux duality that has been the fashion of Linux publishers and FOSS chat groups. Stallman objects to *all* software products that don’t conform to the four freedoms outlined on the Free Software Foundation website [3], which includes many other products in addition to Microsoft products. (For an education on this matter, see the list of Free GNU/Linux distributions at the Free Software Foundation site, which excludes nearly all of the mainstream distros that receive media attention for being “free” –

## Info

- [1] “Free software advocate Richard Stallman spoke at Microsoft Research this week”: <https://www.zdnet.com/article/free-software-advocate-richard-stallman-spoke-at-microsoft-research-this-week/>
- [2] Reasons Not to Use Microsoft: <https://stallman.org/microsoft.html>
- [3] Four Freedoms: <https://www.gnu.org/philosophy/free-sw.html>
- [4] Free GNU/Linux distributions: <https://www.gnu.org/distros/free-distros.html>

including Debian [4].) So Stallman showing up at Microsoft is not exactly like a historic resolution heralding a new era of Pax Britannica. It is yet another visit to yet another software company that could use some enlightenment.

Richard Stallman did not go to Microsoft to negotiate. He is an ambassador for an idea, and he went there to make his case, just as he does all around the world. According to the report at ZDNet, he gave “mostly the standard talk, covering the importance of free software, GPL v3, GNU vs. Linux.”

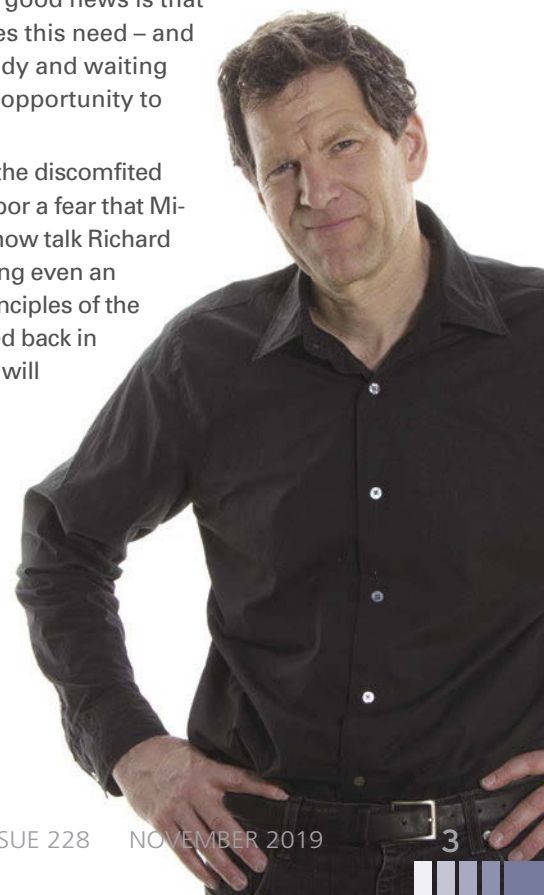
The most interesting (and newsworthy) part of the talk is the new topics, which reveal how the goals of the Free Software Foundation are playing out through a new crop of contemporary questions and problems. Stallman’s list of “small requests” for Microsoft shed light on where we are *now*. For instance, reports are that he suggested Microsoft advocate for more consistent free licensing at the GitHub open source code site, which Microsoft purchased in 2018, and that the company use its clout to force manufacturers to publish their hardware specs, so that developers will have an easier time working their way around the complications of the Microsoft-driven Secure Boot feature of the UEFI standard.

Interestingly, back when Microsoft was a sworn enemy of Free Software, they didn’t really have to listen to anything the Free Software Foundation said. Now that they are supposed to be playing nicely, there is a need for them to engage in dialog. The good news is that Microsoft recognizes this need – and that Stallman is ready and waiting to capitalize on the opportunity to make his case.

So just an FYI to all the discomfited bloggers: If you harbor a fear that Microsoft could somehow talk Richard Stallman into budging even an inch on the clear principles of the movement he started back in 1985, forget that...it will never happen!



Joe Casad,  
Editor in Chief





## WHAT'S INSIDE

**Sure you've heard** of the Tor Browser, but what about OnionShare? This cool tool for the invisible web lets you share files without revealing your identity.

Also in this month's issue:

- **Baobab** – find and remove unnecessary files and directories (page 24).
- **Secure Online Passwords** – discover some tools and techniques for keeping online passwords safe (page 32).

Check out MakerSpace for a look at how to expand the range of wireless communication for IoT devices using the low-power and long-range LoRa communication protocol. Also inside, our LinuxVoice section, we study the Lutris gaming platform and takes a swat at McFly, an innovative solution that brings the power of AI to the Bash shell.

## SERVICE

- 3 Comment
- 6 DVD
- 96 Featured Events
- 97 Call for Papers
- 98 Preview

## NEWS

### 08 News

- The New Kali Linux Is Here
- exFAT Is Coming to Linux
- Knoppix 8.6 Released
- openSUSE Board Gets a New Chairman
- Bluetooth Vulnerability Makes Spying Easy
- Open Source Webmin Had Backdoor for More Than a Year

### 12 Kernel News

- Creating libperf
- Using GCC Extensions
- Editing the Laws of the Universe

## REVIEWS

### 20 Sayonara Player

For a simple audio player, check out Sayonara, a great choice for enjoying all your favorite music, Internet radio, and podcasts.



## COVER STORY

### 16 Anonymous File Sharing with OnionShare

OnionShare lets you share files without revealing IP addresses or domain names. The latest version also allows uploads.



## IN-DEPTH

### 24 Baobab

Unnecessary files and directories take up valuable space on hard disks or SSDs. Baobab lets you locate and remove data garbage at the push of a button.





IN-DEPTH

**26 Charly's Column – ntpviz**  
The Network Time Protocol allows admins to keep time on their computers, but this timekeeping is only moderately successful. Charly uses the ntpviz statistics tool to visualize time fluctuation.

**28 Command Line – Dialog**  
Create dialog boxes with checkboxes, progress bars, and many other features that users may find helpful when working at the command line.



**32 Secure Online Passwords**  
Securely storing passwords online can be a complex task. With a few tools, websites can offer better security, but users still need to choose their passwords wisely.

**36 AppArmor**  
Today's security environment is a tumultuous landscape riddled with threats. AppArmor offers an extra ring of protection for your system, and it is easier to learn and implement than many alternative mandatory access control solutions.

**42 Programming Snapshot – cdbm**  
When you change directories at the command line, you often find yourself jumping back and forth between known paths. With a utility written in Go, Mike Schilli records the jumps and shows the way back.



**46 Debian /usr Merge**  
At long last, Debian joins other Unix-like distributions in merging /usr directories.

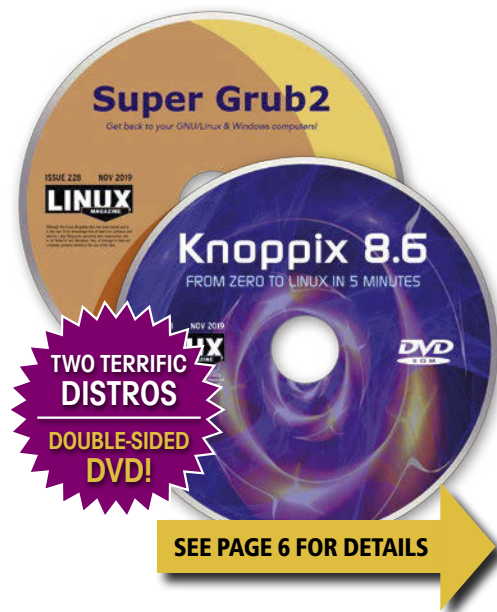
MAKERSPACE

**50 Tasmota**  
Flashing IoT devices with new firmware lets you wield control and keep them out of the cloud.

**56 Long Range Modem**  
WiFi is convenient for devices that are in the same house. If you want to extend the distance, give LoRa a call.

**62 Open Hardware – Keyboardio**  
Jessie Vincent, cofounder of Keyboardio, looks back at the process of designing an efficient and comfortable keyboard.

**66 HiFiBerry**  
Build a networked receiver for your digital music collection with an old stereo, a Raspberry Pi, and the HiFiBerry.



LINUXVOICE

**71 Welcome**  
This month in LinuxVoice.

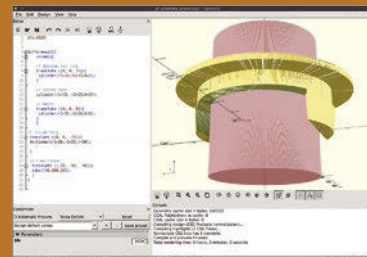
**73 Doghouse – Licensing**  
For entrepreneurs with little money, FOSSH offers a way to get their projects off the ground.

**74 Lutris**  
If you frequently play games on Linux, you are accustomed to dealing with many different installers and configurations. Lutris can help simplify the process of setting up all your games.



**80 McFly**  
McFly extends the Bash history's features and helps you find past commands more quickly.

**84 FOSSPicks**  
As you might guess from certain titles in this month's selection, Graham has finally built himself an open source 3D printer.



**92 Tutorials – Mastodon**  
The open and simple Mastodon API makes it easy to create applications to interact with this federated microblogging platform.



# On the DVD



## Knoppix 8.6

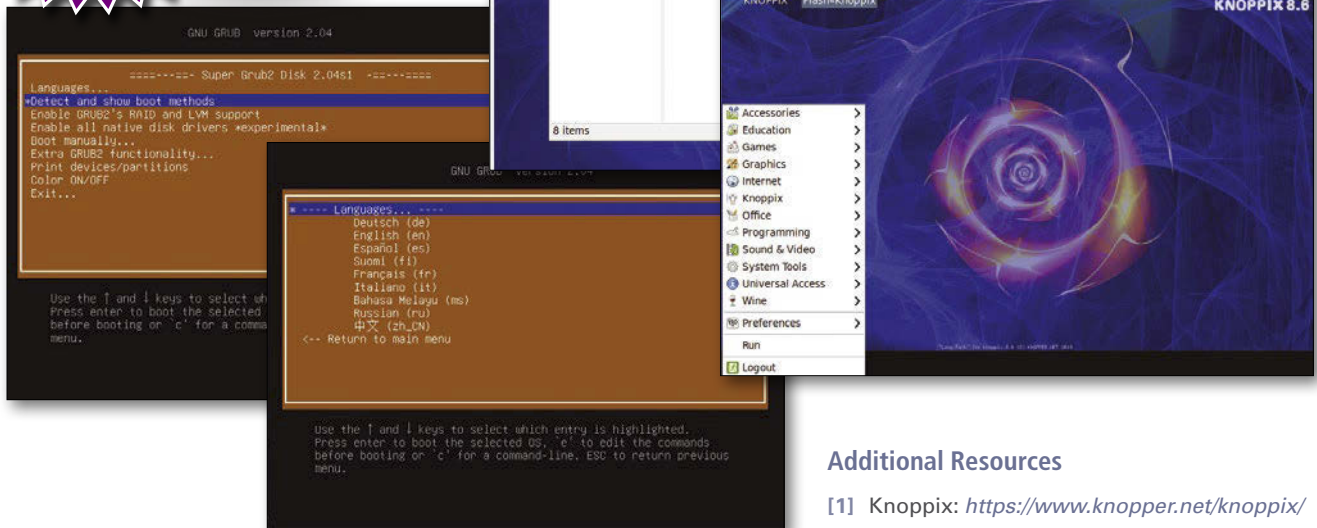
Knoppix is the quintessential Live rescue disk, with a helpful collection of tools for bringing back both Linux and Windows systems. But Knoppix is also much more: Onboard, you'll find a full-featured GUI desktop and over 2,600 packages, including diagnostic tools, development tools, pen-test utilities, and lots of well-known and unknown end-user applications.

## Super Grub2

This specialty Linux builds a menu of operating systems on your computer and lets you choose which system you would like to boot. Super Grub2 is useful for fixing computers that have lost their boot menu and finding old OS versions lost on your hard disk. Other useful features include the ability to change the boot menu language.

TWO TERRIFIC  
DISTROS

DOUBLE-SIDED  
DVD!



## Additional Resources

- [1] Knoppix: <https://www.knopper.net/knoppix/index-en.html>
- [2] Knoppix Info: <https://www.knopper.net/knoppix-info/index-en.html>
- [3] Super Grub2: <https://www.supergrubdisk.org/super-grub2-disk/>
- [4] Super Grub2 Wiki: <https://www.supergrubdisk.org/wiki/SuperGRUB2Disk>

Defective discs will be replaced. Please send an email to [subs@linux-magazine.com](mailto:subs@linux-magazine.com).

Although this Linux Magazine disc has been tested and is to the best of our knowledge free of malicious software and defects, Linux Magazine cannot be held responsible and is not liable for any disruption, loss, or damage to data and computer systems related to the use of this disc.



Shop the Shop

[shop.linuxnewmedia.com](http://shop.linuxnewmedia.com)

# DISCOVER LibreOffice



Explore the **FREE** office suite used by busy professionals around the world!

Create your own:

- Word processing docs
- Spreadsheets
- Presentations
- Databases

Whether you work on a Windows PC, a Mac, or a Linux system, you have all you need to get started with LibreOffice today. This single-volume special edition will serve as your guide!

Order online:

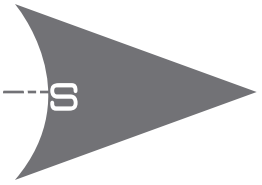
[shop.linuxnewmedia.com/specials](http://shop.linuxnewmedia.com/specials)

For Windows, macOS, and Linux users!



# NEWS

Updates on tech



## THIS MONTH'S NEWS

- 08** • The New Kali Linux Is Here
  - exFAT Is Coming to Linux
- 09** • Knoppix 8.6 Released
  - openSUSE Board Gets a New Chairman
  - More Online
- 10** • Bluetooth Vulnerability Makes Spying Easy
  - Open Source Webmin had Backdoor for More Than a Year

### The New Kali Linux Is Here

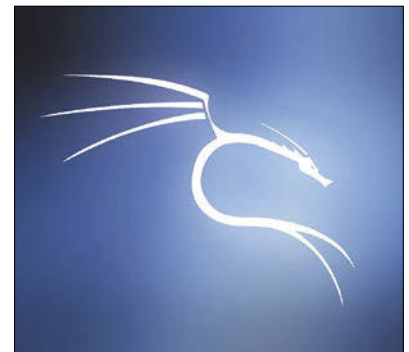
Offensive Security has announced the release of Kali Linux 2019.2 (<https://www.kali.org/releases/kali-linux-2019-3-release/>). Kali Linux is one of the most popular and widely used penetration testing Linux distributions.

The release is based on Linux kernel 5.2.9, and includes new features across the board with NetHunter, ARM, and packages, as well as the normal bug fixes and updates, according to the project's announcement page.

According to Softpedia, "The Kali Linux NetHunter project for running the OS on Android devices has been updated as well in this release with support for new smartphones, including LG V20 International Edition, Nexus 5X, Nexus 10, and OnePlus 7, the latter being Offensive Security's new flagship device for Kali Linux NetHunter."

To speed up delivery of the distro and subsequent updates, Offensive Security has partnered with Cloudflare to use its content delivery network (CDN) to mirror its repositories.

Users can download Kali Linux from the official page.



### exFAT Is Coming to Linux

exFAT is one of the most popular file systems used on external devices like SD cards and Flash drives. Microsoft collects license fees from the vendors that use exFAT in their products.

In an unexpected move, Microsoft made two decisions that make exFAT an "open" (but not open source) file format that anyone can use (<https://cloudblogs.microsoft.com/opensource/2019/08/28/exfat-linux-kernel/>).

First, Microsoft is contributing all exFAT patents to OIN, which will allow its members to use exFAT without any legal threat from Microsoft.

Second, Microsoft has published the technical specification of exFAT that the Linux kernel community can now use to write exFAT drivers.



"It's important to us that the Linux community can make use of exFAT included in the Linux kernel with confidence. To this end, we will be making Microsoft's technical specification for exFAT publicly available to facilitate the development of conformant, interoperable implementations. We

also support the eventual inclusion of a Linux kernel with exFAT support in a future revision of the Open Invention Network's Linux System Definition, where, once accepted, the code will benefit from the defensive patent commitments of OIN's 3040+ members and licensees," said John Gossman, Microsoft Distinguished Engineer and Linux Foundation Board Member.

There already is an exFAT driver available as an external module, which users can install on their systems. Because of patents, the driver could not be included in the kernel. The announcement from Microsoft changes that.

The only issue with the existing exFAT driver is that it is based on code that was leaked from Samsung's implementation of exFAT back in 2013. The kernel community could not touch the code because of patent issues, so it is not currently up to community standards.

Now the kernel community can take a stab at the existing code and clean it for the kernel. Soon, Linux users will have native support for exFAT.

## Knoppix 8.6 Released

Klaus Knopper has announced the release of the latest version of the Knoppix Live GNU/Linux distribution (<https://news.softpedia.com/news/knoppix-live-gnu-linux-system-is-now-based-on-debian-gnu-linux-10-buster-527047.shtml>). Knoppix is a classic Live Linux that is often used to repair and restart downed Linux and Windows systems.

Version 8.6 of Knoppix is based on Debian/stable (buster), with some packages from Debian/testing and unstable (sid) for newer graphics drivers or desktop software packages. Knoppix uses Linux kernel 5.2.5 and Xorg 7.7 (core 1.20.4) for supporting current computer hardware.

Knoppix is suitable for both new and old hardware. "Both 32-bit and 64-bit kernels are included for supporting both old and new computers; the 64-bit version also supports systems with more than 4GB of RAM and chroot to 64-bit installations for system rescue tasks. The bootloader will start the 64-bit kernel automatically if a 64-bit-capable CPU is detected," according to the release notes.

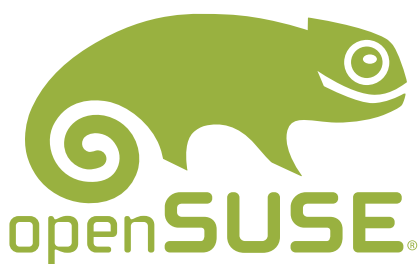
The latest Knoppix comes with a boatload of packages pre-installed, including the GNOME 3 and KDE Plasma 5 desktop environments and the Wine 4.0 compatibility tool for supporting Windows applications.

Knoppix 8.6 is available for free download (<https://www.knopper.net/knoppix/knoppix860-en.html>).

## openSUSE Board Gets a New Chairman

Long-time openSUSE contributor Richard Brown is stepping down from his role as chairperson of openSUSE board, a position he had been holding for the last five years (<https://finance.yahoo.com/news/gerald-pfeifer-appointed-chair-open-suse-195800897.html>). He will be replaced by Gerald Pfeifer, SUSE's CTO for EMEA. Gerald himself is a developer who has contributed to projects like GCC and Wine.

In a blog post, Brown said, "Some of the key factors that led me to make this step include the time required to do the job properly and the length of time I've served. Five years is more than twice as long as any of my predecessors. The time required to do the role properly has increased, and I now find it impossible to



## MORE ONLINE

### Linux Magazine

[www.linux-magazine.com](http://www.linux-magazine.com)

### Linux Administration Focus

<http://www.linux-magazine.com/tags/view/administration>

#### Restoring Deleted Files in Linux • Ken Hess

If you thought that you couldn't restore deleted files in Linux, you didn't get the whole truth. The truth will set you free and possibly recover those deleted files.

#### FOG Project • Mayank Sharma

Use the FOG imaging server to image and rollout several installations with a single click.

### ADMIN HPC

<http://www.admin-magazine.com/HPC/>

#### High-Performance Python – Compiled Code and Fortran Interface • Jeff Layton

Fortran functions called from Python bring complex computations to a scriptable language.

### ADMIN Online

<http://www.admin-magazine.com/>

#### A Self-Healing VM System • Frank H. Walden

The right combination of mostly free automation and monitoring tools can create a self-healing system, in which your servers fix themselves.

#### Accelerate Web Applications with Varnish Cache • Stefan Wintermeyer

If a web application delivers its pages too slowly, users quickly move on. Varnish Cache lets you hitch more horses to the cart.

#### How to Back Up in the Cloud Martin Loschwitz

In cloud computing practice, backups are important in several ways: Customers want to secure their data, and vendors want to secure the essential details of their platforms. Rescue yourself, if you can.

balance the demands of the role with the requirements of my primary role as a developer in SUSE, and with what I wish to achieve outside of work and community.”

Brown will focus on his work at SUSE’s Future Technology Team that works on emerging technologies.

“I could not be more excited and humbled to participate in the openSUSE Project as board chair,” Pfeifer said. “Collaboration in the openSUSE community has contributed to remarkable Linux distributions, and I’m looking forward to ongoing growth in both the community and the openSUSE distributions – Linux and beyond – and tools. openSUSE is at the leading edge of a historic shift, as open source software is now a critical part of any thriving enterprise’s core business strategy. This is an exciting time for the openSUSE community, as well as for open source at large.”

The openSUSE project is funded by SUSE, but it is a community driven project where decisions are made by the community. The openSUSE distros are also upstream to many SUSE products, such as SUSE Linux Enterprise and SUSE CaaSP.

## Bluetooth Vulnerability Makes Spying Easy

Bluetooth is one of the weakest links that opens doors for attacks. A newly discovered vulnerability in Bluetooth enables bad actors to spy on data flowing between two devices (<https://thehackernews.com/2019/08/bluetooth-knob-vulnerability.html>).

According to the Hacker News, “The vulnerability, assigned as CVE-2019-9506, resides in the way ‘encryption key negotiation protocol’ lets two Bluetooth BR/EDR devices choose an entropy value for encryption keys while pairing to secure their connection.”

The vulnerability exposes billions of smartphones, laptops, and industrial devices.

There is nothing users can do to protect themselves at this time. According to an advisory by Carnegie Mellon University (<https://www.kb.cert.org/vuls/id/918987>), Bluetooth host and controller suppliers should refer to the Bluetooth SIG’s “Expedited Errata Correction 11838” for guidance on updating their products. Downstream vendors should refer to their suppliers for updates.



Image © Alexander Sidorov, 123RF.com

## Open Source Webmin had Backdoor for More Than a Year

Webmin developer’s have disclosed the critical zero-day vulnerability found last week wasn’t a flaw; it was planted by a hacker.

Someone planted a backdoor into the build infrastructure of Webmin, and it remained undetected through version 1.882 to 1.921.

Researcher Özkan Mustafa Akkuş who discovered the vulnerability, did not inform the project about the backdoor and publicly disclosed it at DefCon (<https://thehackernews.com/2019/08/webmin-vulnerability-hacking.html>).



Image © Bruce Rolff, 123RF.com

Joe Cooper, one of Webmin’s developers, called it an unethical practice, giving the project no time to work on a fix to protect users.

Akkuş also released a Metasploit module to exploit the vulnerability.

Webmin developers fixed the flaw by removing the backdoor. Webmin is a popular open-source web-based application for managing Unix-based systems.





# Join us at DrupalCon Amsterdam!

REGISTER NOW ON  
[EVENTS.DRUPAL.ORG/AMSTERDAM2019](https://events.drupal.org/amsterdam2019)

DrupalCon Amsterdam 2019 will take place **from 28 – 31 October at the RAI Amsterdam**, with high quality Keynote Speakers like **Dries Buytaert, Dr. Sue Black and Boris Veldhuijzen van Zanten**



# Zack's Kernel News



Chronicler Zack Brown reports on the latest news, views, dilemmas, and developments within the Linux kernel community.

By Zack Brown

## Author

The Linux kernel mailing list comprises the core of Linux development activities. Traffic volumes are immense, often reaching 10,000 messages in a week, and keeping up to date with the entire scope of development is a virtually impossible task for one person. One of the few brave souls to take on this task is **Zack Brown**.

## Creating libperf

Jiri Olsa recently posted a large patchset to begin the process of migrating the perf profiling code out of the core kernel code and into its own *libperf* library. perf is a debugging tool that is virtually never encountered by regular users. It profiles parts of the kernel in order to identify bottlenecks and other slowdowns. This lets the developers know which areas of the kernel might offer a big reward for receiving their attention.

Converting perf to a library is certainly a large task, so Jiri's code was just an initial pass at creating a library infrastructure (based largely on the existing *libbpf* code), onto which more and more could be migrated over time. Initially, the code was limited to basic counting operations, such as tallying up the number of CPUs and threads or enabling and disabling events. In the perf world, an "event" is a trigger point that allows perf to do various actions in the midst of a piece of kernel code that has no idea it's being profiled.

The problem with creating a full *libperf* library to entirely replace perf in one fell swoop is that it's too prone to errors. Jiri said the amount of code that would need to be rewritten was truly vast, so the likelihood of creating lots and lots of bugs was pretty high. Creating the stable basic infrastructure first, and then migrating the various pieces in a relatively straightforward progression, would avoid that problem and would make the bug-hunting process much cleaner for everyone.

Jiri's plans for the future were to start adding ways to actually use the collected data and event handling that had been part of the initial pass. Eventually the code would also migrate away from the perf default directory into a new directory more appropriate for a public library.

Ian Rogers from Google was thrilled with this whole project. His team had been working on methods of converting perf data to other forms, to avoid overhead – presumably because they were running perf on production systems that

needed to be sleek and fast yet still report problems to the sys admin teams.

Some of Ian's concern was interface cleanliness and compatibility with C++, which was the language they mostly used on his team. Jiri replied that with a little more work, *libperf* would soon offer some higher-level interfaces that Ian might find simple and useful for his needs.

Song Liu also replied to Jiri's initial post, pointing out that the event code in Jiri's initial patchset was really an abstraction rather than a full interface. He added that there were a lot of tools that currently used perf that didn't rely on those abstractions. He said, "I am not sure whether these tools would adopt *libperf*, as *libperf* changes their existing concepts/abstractions."

Jiri replied that *libperf* would eventually include the rest of the perf API, so those other tools would not be forced to adapt abstractions that didn't match their use cases.

Arnaldo Carvalho de Melo also replied to Song, reiterating Jiri's point, saying, "for now, we're just trying to have something that is not so tied to perf and could possibly be useful outside `tools/perf/` when the need arises for whatever new tool or preexisting one. There are features there that may be interesting to use outside perf; time will tell." Arnaldo added that Jiri "is just slowly moving things to a public *libperf* while keeping perf working; in the end, the goal is to have as much stuff that is not super specific to some of the existing perf tools (`tools/perf/builtin-*.c`) in *libperf* as possible. It is still early in this effort; that is why he is still leaving it in `tools/perf/lib/` and not in `tools/lib/perf/`."

Song replied that he liked this strategy and admired the amount of work it would take.

Arnaldo also replied to Jiri's initial patchset, saying, "I've tested it in various distros and made fixes in the relevant csets to avoid breaking bisection; it builds everywhere I tested so far, except on Fedora Rawhide, but that is something



unrelated, a coincidence since I refreshed that container yesterday (one Python hiccup and something else); I've made some changes to the docs adding some articles and adding some clarification about refcounts not necessarily destroying the object, just dropping a reference, pushed everything to `tmp.perf/core`, and will do the whole container testing soon."

And Alexey Budankov from Intel offered his approval of the whole patchset as well. He remarked, "Some API for reading perf record trace could be valuable extensions for the library. Also at some point public API will, probably, need some versioning."

There was no reply to that, and the conversation came to an end. Personally I love seeing the progression of these various tools. There are so many of them! Something is needed for the Linux kernel. Someone creates it (or more often, a lot of people try to create it, and something coherent gradually emerges). Eventually, someone sees that it could have a wider value than just for the Linux kernel, and, at some point, it is abstracted out of the core kernel code and made into a standalone thing that helps people throughout the world.

## Using GCC Extensions

Interactions between the kernel source tree and the GCC compiler are almost always strange. Their incestuous inter-twinings and rivalrous collaborations have forced their respective developers to deal with many harsh truths, concerning which favorite feature is truly to be determined by one project or the other.

In this particular case, Joe Perches wanted the Linux kernel to support GCC's `fallthrough` attribute, introduced in GCC v7. GCC attributes are special hints that can appear directly in your code, but instead of being part of the C or C++ language, they are interpreted specially by GCC, to help it produce the absolute best possible machine code in your compiled binary.

The `fallthrough` attribute, for example, is used in `switch` statements, to indicate to GCC that a given "case" is going to be allowed to fall through to the enclosing block, rather than jumping anywhere else.

The justification is clear – GCC doesn't want to extend the C language, because the goal is not to produce more code, but

to prevent code from being produced. The goal of these linguistic extensions is to avoid the need to produce machine code. So, instead of extending the language directly, GCC allows users to insert these non-C-language hints into perfectly good C code.

Joe posted patches to update the kernel source tree, to reserve all 4,200 occurrences of the word "fallthrough" as a "pseudo keyword," so it would only appear in the source tree in places where it was actually intended to be used as a GCC attribute. The kicker is that one of the ways to use these attributes is not as text inserted directly into the code, but as a standard C comment, bounded by `/*` and `*/`.

Peter Zijlstra burst into applause. And Pavel Machek immediately approved the patch for inclusion in the kernel. Pavel also asked if the "fallthrough" C comment would also be recognized by GCC if it appeared in a macro; Joe replied that GCC would, but the non-GCC code checkers and other tools probably would not.

Kees Cook was also concerned that he did not want to break existing code scanners, especially the Coverity scanner. He said, "I'd like to make sure we don't regress Coverity most of all. If the recent updates to the Coverity scanner include support for the attribute now, then I'm all for it." But to this, Peter replied, "Coverity can go pound sand; I never see its output, while I get to look at the code and GCC output daily."

Miguel Ojeda also pointed out that this entire topic had been raised in the past, when the conclusion was that it would be better to wait until the ecosystem of surrounding tools supported the attribute and wouldn't be broken by the change. He asked, "Is everyone happy this time around?"

Joe replied to Miguel, saying that in fact, his patches didn't actually change the kernel to use the `fallthrough` attribute – they only reserved the word "fallthrough." So, he said, "Patches that convert `/* fallthrough */` et al. to `fallthrough` would only be published when everyone's happy enough."

Meanwhile, H. Peter Anvin asked what would happen if someone ran the kernel source tree through a comment stripper before compiling – specifically, could the comments be replaced with some other "magic token" to accomplish



the same thing. This led to a technical discussion surrounding exactly how these attributes should be represented in the kernel, to work best with the various existing tools. Should it be a comment? Should the word “fallthrough” have underscores attached at either end? And so on. At one point Joe asked Linus Torvalds to make a determination, and Linus replied:

*“My only real concern is that the comment approach has always been the really traditional one, going back all the way to lint days.*

*“And you obviously cannot use a #define to create a comment, so this whole keyword model will never be able to do that.*

*“At the same time, all the modern tools we care about do seem to be happy with it, either through the GCC attribute, the Clang `[[clang:fallthrough]]`, or the (eventual) standard C `[[fallthrough]]` model.*

*“So I’m ok with just saying ‘the comment model may be traditional, but it’s not very good.’”*

The discussion continued a short while, essentially with implementation details. The real issue at the root of GCC/kernel interactions is that each project sees itself as more fundamental than the other. The GCC developers feel GCC is more fundamental, because it is responsible for building all software in the solar system, not just Linux; while the kernel developers feel the kernel is more fundamental, because it is responsible for running all hardware in the solar system. The GCC people aren’t going to love the idea of implementing a feature just to make the kernel developers’ lives easier; while the kernel people aren’t going to love the idea of having to rely on GCC features that dictate how the final machine code will end up.

In the past, this debate led Linus to rely on an older and “better” version of GCC for a very long time, even as the GCC code continued to grow and develop. In some ways, it was the debate to end all debates, and now the GCC developers and kernel developers keep in better contact and have a friendlier relationship.

## Editing the Laws of the Universe

Lately, the Linux kernel developers have been rewriting the core scheduler code. I

repeat: the core scheduler code. This is the part of the kernel that decides how and when to switch between running processes. Process switching generally happens so rapidly that you can have tons of users all logged into your system at the same time, and all have a smooth, pleasant experience. It’s what makes our computers “multitasking” instead of “single-tasking.”

It’s also notoriously difficult to test. How can you tell if one core scheduler implementation is better than another? Or for that matter, how can you tell if a single patch improves an existing core scheduler or makes it choppier or slower? The core scheduler is supposed to work well on billions of computers, including virtual systems, running on every conceivable hardware configuration, for users engaged in any conceivable set of use cases – not just porn.

In fact, there’s really no way to perform exhaustive and correct tests. Developers working in the area of the core scheduler just basically ... do their best. They think really hard. They invoke the muses. And they do their best to generate convincing explanations that fit into simple paragraphs in a changelog entry.

Vineeth Ramanan Pillai (on behalf of many co-developers) recently posted the latest iteration of patches to rewrite the core scheduler. In this iteration, the code was mostly concerned with getting the basic ideas right, avoiding crashes, and running fast. Among other requirements, virtual systems and CPU hot plugging took center stage.

Aubrey Li posted with some test results, including problems with the tests themselves in this new version of the scheduler; Julien Desfossez and Aaron Lu immediately jumped in to help debug the tests.

At one point, Julien offered his assessment of these latest patches from Vineeth, saying, “it helps for untagged interactive tasks and fairness in general, but this increases the overhead of core scheduling when there is contention for the CPU with tasks of varying CPU usage. The general trend we see is that if there is a CPU-intensive thread and multiple relatively idle threads in different tags, the CPU-intensive tasks continuously yield to be fair to the relatively idle threads when it becomes runnable. And if the relatively idle threads make up for

most of the tasks in a system and are tagged, the CPU-intensive tasks see a considerable drop in performance.”

After finding the problem with Aubrey’s testing scripts, Aubrey posted some new benchmarks, explaining that at his job, “The story [that] we care about latency is that some customers reported their latency critical job is affected when co-locating a deep learning job (AVX-512 task) onto the same core, because when a core executes AVX-512 instructions, the core automatically reduces its frequency. This can lead to a significant overall performance loss for a non-AVX-512 job on the same core.”

With the new patches, Aubrey reported improved results in these tests.

Meanwhile, Julien, from the dark depths, remarked, “After reading more traces and trying to understand why only untagged tasks are starving when there are CPU-intensive tasks running on the same set of CPUs, we noticed a difference in behavior in `pick_task`. In the case where `core_cookie` is 0, we are supposed to only prefer the tagged task if its priority is higher, but when the priorities are equal we prefer it as well, which causes the starving. `pick_task` is biased toward selecting its first parameter in case of equality, which in this case was the `class_pick` instead of `max`. Reversing the order of the parameter solves this issue and matches the expected behavior.”

Subhra Mazumdar also posted some benchmark tests, showing good results for database use cases. Subhra suggested that the particular configuration that produced this result should be made standard by default in the core scheduler. Julien replied that yes, this was not even going to be optional, but would just be the standard behavior in later versions of the scheduler.

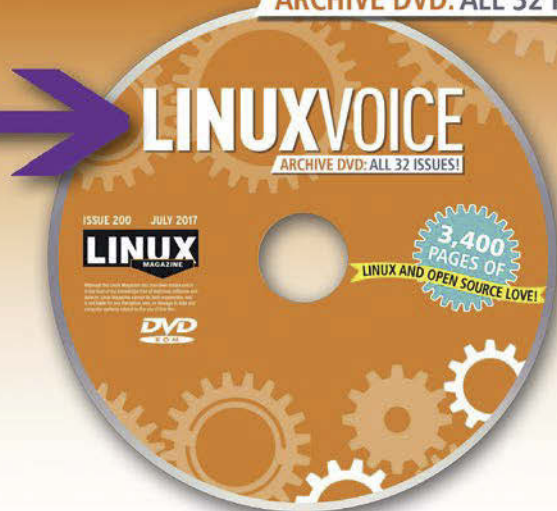
Aubrey also reported some new benchmarks, in which a set of virtual systems appeared to suffer from unfair scheduling under certain circumstances. Unfairness refers to some processes getting more CPU time than others, or some CPUs being used more than others. Julien confirmed that this was a reproducible case and began looking for what might have caused it. Tim Chen was also interested in tracking this down and started hacking at the code to see what could be discovered.

This debugging session continued, with more tests and patches flying around, and the developers just generally having the time of their lives. More developers joined the fun, such as Dario Faggioli from SUSE, and it was all magic.

Nothing whatsoever was decided, accepted, rejected, or anything like that. What we had here was a bunch of people not caring in the slightest whether the gaze of history was on them and just playing the music of the universe with nothing but love. It was wonderful just to get to listen. ■■■

# THE COMPLETE LINUXVOICE

ARCHIVE DVD: ALL 32 ISSUES!



3,400  
PAGES OF  
LINUX AND OPEN SOURCE LOVE!

Since April 2014, Linux Voice has showcased the very best that Free Software has to offer. Now you can get it all on one searchable DVD.

**Includes all 32 issues in  
EPUB, PDF, and HTML format!**

**Order now!**

[shop.linuxnewmedia.com](http://shop.linuxnewmedia.com)



## Anonymous File Sharing with OnionShare 2.0

# Secret Files

OnionShare lets you share files without revealing IP addresses or domain names. The latest version also allows uploads. *By Christoph Langner*

The Tor Project [1] has spawned a whole global community dedicated to the concept of anonymous browsing. The project's Tor Browser [2], which lets the user surf the web without leaving a trail or trace, was originally intended to help dissidents in totalitarian countries communicate without surveillance, but since then, it has become popular with whistle-blowers, drug buyers, and millions of everyday users who simply don't want to submit to the culture of tracking and targeting that exists on the mainstream Internet.

The core technology behind the Tor Browser is a technique known as *onion routing*. Onion routing routes a message through a message of participating routers that could be anywhere on the Internet. A data packet takes a random path through a series of the routers. The packet is encapsulated in multiple layers of encrypted routing information. Each router receives the packet and uses its private key to decrypt the outermost layer, which contains a destination address for where to forward the packet, and then sends the data on to the next link in the chain.

The power of onion routing is that no single router on the network has complete knowledge of where the packet came from and where it is going. Each router can only read the single layer specifically encrypted and addressed to it. This technique is known as onion routing, because the many layers of encrypted routing information resemble the layers of an onion that are gradually peeled away as the packet makes its way through the network. (For more information, see the "How Private?" box.)

The Tor Browser is a standard tool at this point that is well known to many Linux users, and tutorials on how to use Tor have appeared in many forms – including in this magazine.







## How Private?

The onion routing process is considered relatively secure, although various attack scenarios are known [4]. One of the points of criticism lies in the inexperience of the users. To really protect your privacy, you need to do more than simply route your traffic through Tor. You also have to harden the browser and preferably the whole system. For this reason, the project offers the preconfigured Tor Browser with an integrated Tor client for all common operating systems. As an alternative, the developers are also working on Tails [5], a live distribution that offers an inherently secure platform.

Tor comes with the Onion Service Protocol [6] and the hidden services that help users not only use, but also to provide, services anonymously.

Like so many things, hidden services have two sides. Although anonymous routing allows criminal machinations on the one hand, on the other, it genuinely helps people to protect their privacy. Hidden services can also be used to exchange files without participants having to reveal IP addresses, domain names, or account details.

Less well known is OnionShare [3], a useful tool that lets you anonymously share files on Tor networks. Even users who are not interested in long-term participation in the Tor community have learned that OnionShare can be an easy and secure way to post a file without the need for commercial cloud services.

## Anonymous File Sharing

Version 2.0 of OnionShare was released earlier this year and is not yet available in the package sources of the major distributions. Even the brand-new Ubuntu 19.04 “Disco Dingo” runs version 1.3.2 of the program. On the homepage, however, the developers point users to an Ubuntu PPA that supports the installation of the latest version with just a few commands (Listing 1). On Arch Linux, you will find OnionShare in the AUR of the same name. If you can’t find a suitable package for your distribution, see the instructions online for tips on building the current version from scratch [7]. (For a CLI variant, see the “OnionShare as a Service” box.)

After you install, OnionShare will appear in the application menu of the desktop environment. At startup, the program automatically connects to the Tor network. The user interface is very simple. At the top there are two tabs named *Share Files* and *Receive Files*. The settings can also be opened via the gear-wheel icon. The arrow below expands information about the history in a sidebar.

To share files, drag the desired files from the file manager into the application window or use the *Add* button at

## Listing 1: Installing on Ubuntu

```
$ sudo add-apt-repository ppa:micahflee/ppa
$ sudo apt update
$ sudo apt install onionshare
```

## Listing 2: At the Command Line

```
$ onionshare File1 File2
OnionShare 2.0 | https://onionshare.org/
Connecting to the Tor network: 100% - Done
Setting up onion service on port 17607.
Compressing files.
* Running on http://127.0.0.1:17607/ (Press CTRL+C to quit)
```

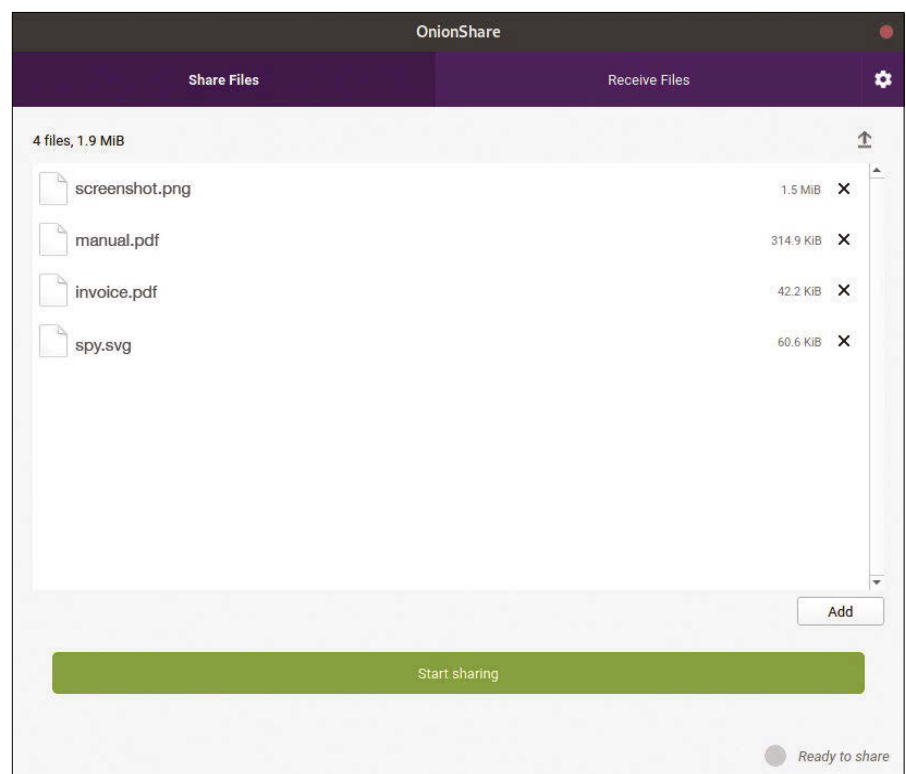
## OnionShare as a Service

On Linux, OnionShare installs two executable files. `onionshare-gui` calls the version with the graphical environment. If you want to run OnionShare on a server without a desktop, start the program by typing `onionshare file_name` or `-` to receive data – by typing `onionshare --receive`. The CLI variant supports all functions of the desktop version (Listing 2).

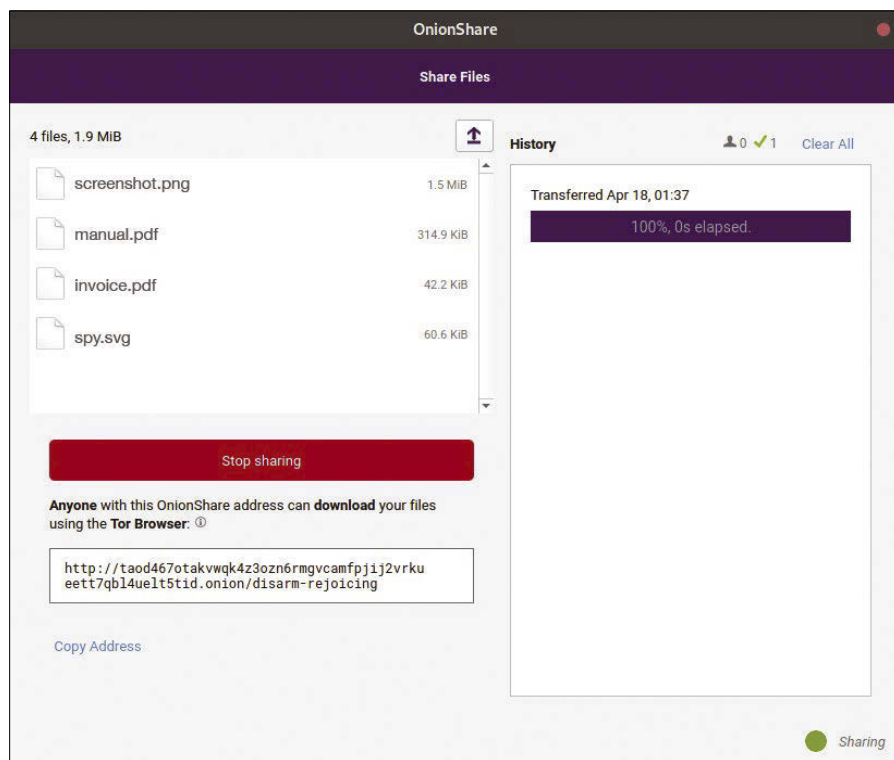
the bottom of the window to open a selection dialog. Once you have added one or more files, start the service by pressing the green *Start sharing* button (Figure 1).

## Download via Browser

To send data to a contact, communicate the OnionShare address now displayed in the window in the style of `http://Tor_address.onion/slug` (Figure 2).



**Figure 1:** File sharing made easy: Drag files into the window, click on *Start sharing*, and send the URL to the contact. Configuration of the router is not required.



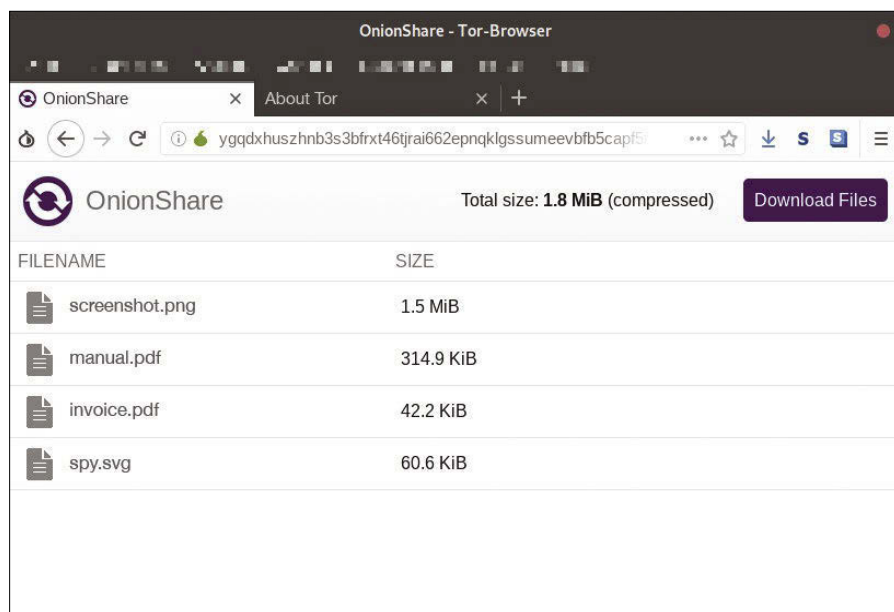
**Figure 2:** Send the OnionShare address to your contact. The history shows how often and when a file was downloaded – but not by whom.

The Tor address is a string of characters assigned to you by Tor (as shown in Figure 2). The slug consists of a random combination of two words at the end of the address, adding an additional layer of complexity to resist guess attacks.

You can share the information either through a secure chat or some manual method. Note that the OnionShare address changes each time the application is started, as long as you do not enable a *persistent address* in the settings. The text cannot be selected directly in the window; to copy it to the clipboard, click *Copy Address*. Port forwarding or further

configuration of the WiFi router are not usually required. If, for example, you would like to offer whistleblowers a portal for sending data in the scope of your journalistic work, you can also publish the address on your homepage.

Uploading data does not differ much from downloading. The Tor Browser acts as the client again. Instead of the list of offered files, you will see an almost empty page. A click on *Browse...* opens a file browser where you can select the data to be uploaded. On pressing *Send Files*, the browser then transfers the file. On the OnionShare program page, you will see the transferred files arriving. After the transmission has been completed, terminate the service by clicking on *Stop Receive Mode*. By default, the program stores the data below `OnionShare/` in the user's home directory.



**Figure 3:** To download data shared via OnionShare, you need the Tor Browser or an active Tor client on your own computer.

configuration of the WiFi router are not usually required.

Your contact does not need a special program to download the data. All they need is the Tor Browser, which is available for all common operating systems or any browser with a Tor client enabled in the system (Figure 3). After you press *Download Files*, the shared data ends up as a ZIP archive on the user's hard disk. However, version 2.0 of the OnionShare client no longer bundles individual files in an archive. In the test, downloading the files also worked on an Android smartphone connected to the Tor network with Orbot [8].

### Receiving Anonymous Data

OnionShare 2.0 not only allows users to share files anonymously but also to receive them anonymously [9]. To receive files anonymously, switch to the *Receive Files* tab and activate Receive mode by pressing *Start Receive Mode*. You will again see an OnionShare address, similar to the one you received when you sent the mail; you have to give this address to

### OnionShare Settings

In most cases, no further configuration is required. In certain scenarios, however, you can check out the settings below the gear icon (Figure 4). For example, to run a permanently active OnionShare server, you need to enable *Public mode* and *Use persistent address*.

Public mode disables a measure that is actually fairly well established as a security function (see the "Slug Protection" box). The second option tells OnionShare to always use the same address. This setting affects privacy but does let you make a share or upload option accessible via OnionShare for a longer period of time. Setting both options is



## Slug Protection

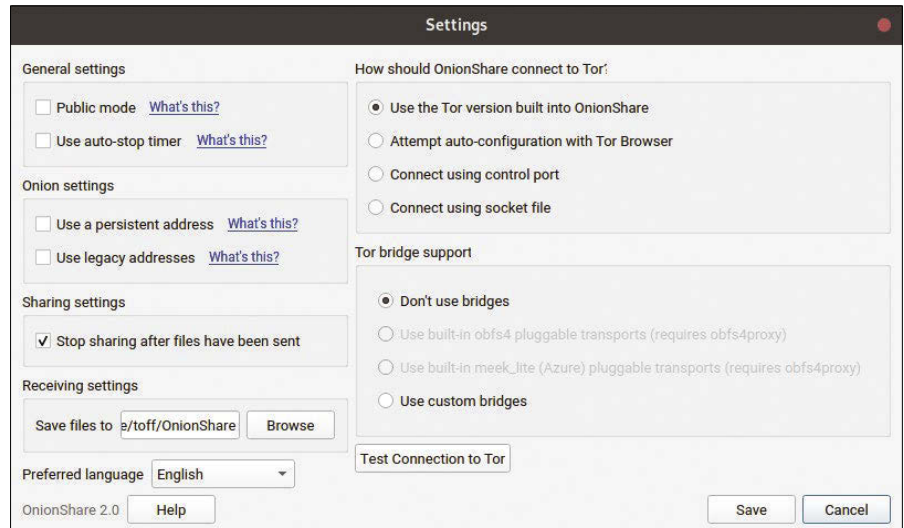
In theory, an attacker could get to an OnionShare by simply trying out Onion addresses. The slug adds another hurdle. If the attacker tries to guess the attachment, OnionShare locks the door completely after a few unsuccessful attempts. However, attackers could use this as a DoS attack and kick OnionShare servers off the network in a targeted manner. To prevent this, the function can be completely deactivated via *Public mode* if required.

therefore recommended if you are setting up an upload server.

But if you want to provide data as a user, and only for a short time – or only for one person – the *Use auto-stop timer* and *Stop sharing after files have been sent* options can help. If you activate the timer, OnionShare will display a dialog when starting the service stating *Stop the share at*; you can now set a time at which OnionShare will automatically remove the share from the network. Alternatively, the program disables sharing as soon as a user has downloaded the files.

## Conclusions

OnionShare is a practical tool even if you are not seeking protection from persecution as a whistle-blower. Without accounts, without commercial cloud services, and without any additional router configuration, you can easily exchange data with other network participants. Your own IP address remains hidden, as do the IP addresses of your contacts. The Tor client included in OnionShare and the Tor Browser with its privacy optimizations ensure that even inexperienced users will receive protection. ■■■



**Figure 4:** You can configure OnionShare via the settings. Most important are the options below General settings and Onion settings.

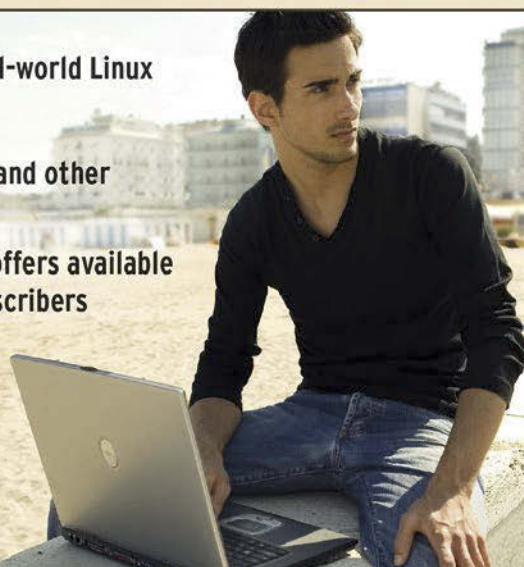
## Info

- [1] Tor Project: <https://www.torproject.org>
- [2] Tor Browser: <https://www.torproject.org/download/>
- [3] OnionShare: <https://onionshare.org>
- [4] Criticism of Tor: [https://en.wikipedia.org/wiki/Tor\\_\(anonymity\\_network\)#Weaknesses](https://en.wikipedia.org/wiki/Tor_(anonymity_network)#Weaknesses)
- [5] Tails: <https://tails.boum.org>
- [6] Tor Onion Service Protocol: <https://2019.www.torproject.org/docs/onion-services.html.en>
- [7] OnionShare building instructions: <https://github.com/micahflee/onionshare/blob/master/BUILD.md#gnulinux>
- [8] Orbot for Android: <https://play.google.com/store/apps/details?id=org.torproject.android&hl=de>
- [9] Changes in OnionShare 2: <https://github.com/micahflee/onionshare/releases/tag/v2.0>

# LINUX UPDATE

Need more Linux? Our free Linux Update newsletter delivers insightful articles and tech tips to your mailbox twice a month.

- Original articles on real-world Linux
- Linux news
- Tips on Bash scripting and other advanced techniques
- Discounts and special offers available only to newsletter subscribers





## Exploring the Sayonara audio player



# Easy Listening

For a simple audio player, check out Sayonara Player, a great choice for enjoying all your favorite music, Internet radio, and podcasts. *By Anzela Minosi*

**D**espite the popularity of Spotify, Deezer, or Apple Music, many music lovers prefer to listen to their own track collection instead of streaming. Sayonara Player helps you organize your music collection, but also integrates Internet radio and podcasts.

Audio players that focus on the essentials, and more importantly still work despite their leanness, are not common on Linux. This makes Sayonara Player [1] all the more surprising. It owes its speed to the fact that it was written in C++. The interface uses the Qt framework, which has a positive effect, because it does not appear overloaded. The simple audio player can be found in the package sources of many distributions and can be installed there using the respective package manager,

### Listing 1: Installation

```
## Installation on Ubuntu:
$ sudo apt-add-repository ppa:lucioc/sayonara
$ sudo apt update
$ sudo apt install sayonara
### Installation on Fedora:
# dnf install sayonara
```

for example, on Ubuntu and Fedora 29 (Listing 1).

### Trivia

To play music, first load it with *File | Open File* or, in the same submenu, with *Open Directory*. The songs you add appear bottom left in the playlist (Figure 1). If you press the button with the “d” (for dynamic load) below the playlist, Sayonara will empty the list automatically when loading new songs. If you click on the *Add* button, the player will add the new songs to the playlist.

You can also save the playlist and load it automatically the next time you start the program. Right click on the tab and then on *Save as* in the context menu. Sayonara then assigns a name to the playlist. Below *File | Settings | Playlist* enable *Load saved playlists*. The next time you start the program, you will see the saved playlists.

Sayonara plays all the popular audio formats such as OGG, WAV, MP3, or the lossless FLAC. Under the hood, the program uses the GStreamer multimedia framework. However, Sayonara cannot play audio CDs; you first need to rip the data carrier to

your hard disk with an audio grabber such as Sound Juicer [2]. Sound Juicer automatically detects the inserted audio CD and saves the songs in Ogg Vorbis format or as MP3 after a click on *Read*. The songs you have read out can then be stored in the playlist.

Sayonara comes with an integrated conversion tool that you can call via *View | Audio Conversion*. You can then use the tool to convert FLAC or WAV files to MP3 or OGG.

The songs can also be organized. Right-clicking on a song in the playlist takes you to the context menu. When you get there, you can call up the lyrics for the current song with the *Lyrics* feature. The *Info* feature displays information about the file, and the cover can be viewed or replaced. Clicking on the *Edit* tab and then on the *Cover* tab takes you to the corresponding settings.

If you check the *Replace* option, you can download several suitable alternatives by clicking on the cover you want to replace, from which you can then select one. In the *Tags* tab you have the possibility to specify the title, the artist, and the album, if Sayonara fails to find any entries on the Internet. In addition, a genre for the song can be assigned here (Figure 2).

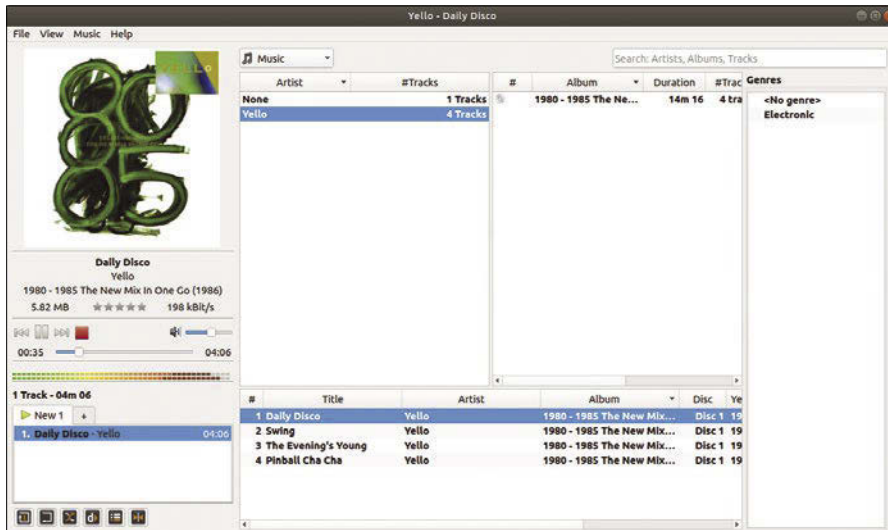


Figure 1: Tracks can even be rated in Sayonara.

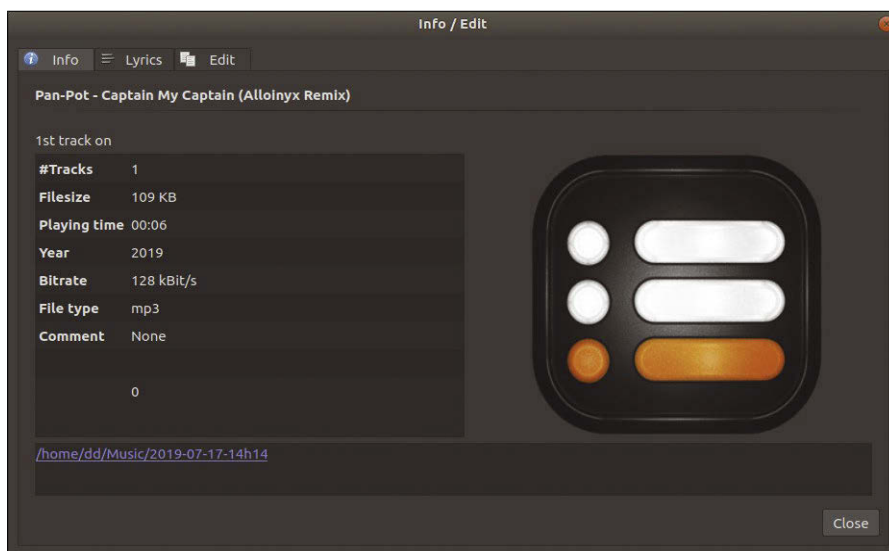


Figure 2: Sayonara comes with an integrated ID3 tag editor that lets you change metadata, such as the artist, title, or genre.

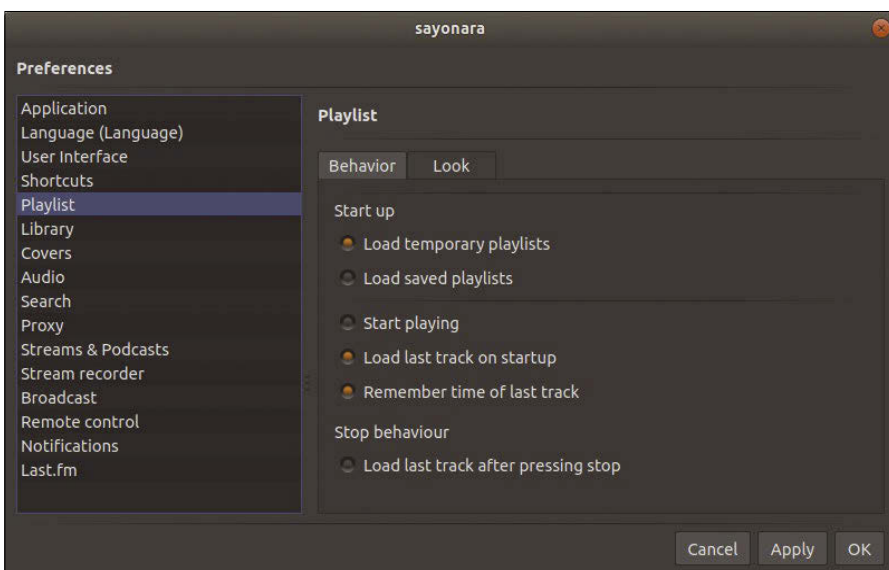


Figure 3: The settings let you customize many details of the player, such as the appearance of the playlist here.

In a song's context menu, you will also find the *Rating* feature, which you can use to assign up to five stars to the current song. To then display additional information in the playlist, such as the rating or cover, open *Settings* and switch to the *Playlist* tab (Figure 3). You can show the rating and the cover with the *Looks* tab. If necessary, enable the *Display Empty Button* option. With the help of this button, which appears above the playlists, you can empty the current playlist with a single mouse click.

## Special Effects

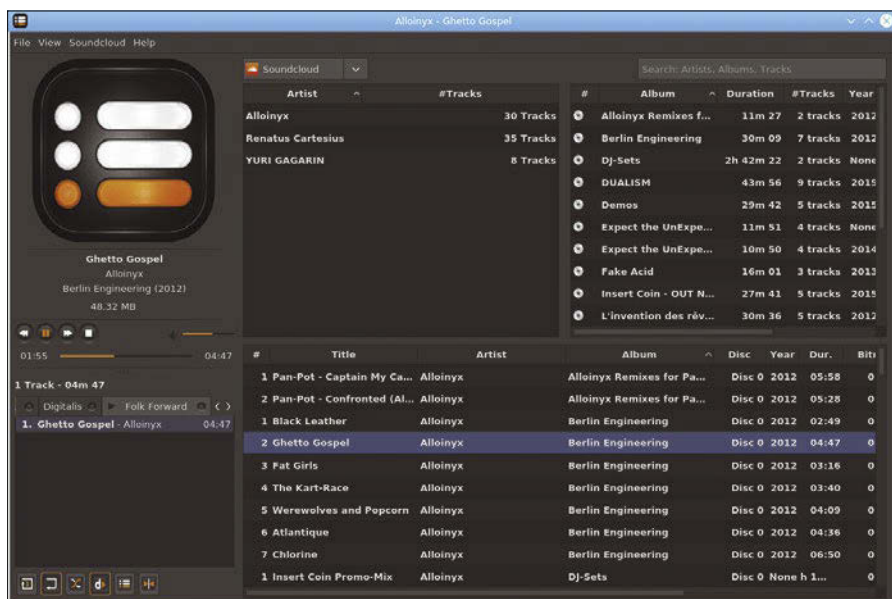
In the settings, you can also specify whether Sayonara should hide in the system tray when you click on the window's close icon. The required settings are *Close to tray* and *Display tray in system* and are located below *File | Settings | Application*.

In the language settings below the *Language* tab, you can choose to operate Sayonara in the language of your choice. Icons, various font settings, and the application's background color can be changed in the *User Interface* tab. If Sayonara cannot play a sound, try switching the sound to PulseAudio or ALSA in the *Audio* tab. Usually, however, the *Automatic* option should suffice.

If you have configured a remote control on your Linux system, it can also be used with Sayonara. The corresponding setting can be found in the *Remote control* tab. If you select *Notifications* from the *Active* tab, Sayonara displays an info box each time you change songs. If you are one of those people who likes to fall asleep with music, it is a good idea to set *File | Shut-down* to have Sayonara automatically shut down the PC at a certain time or after playing the last song in the playlist.

Sayonara visualizes either a level or a spectrum according to the music. These features can be found in the main View menu. The equalizer, which can also be called up there, lets you improve the sound using filters, for example by amplifying the bass.

The transitions between songs in a playlist can be set below *View | Cross-fader*. *Seamless playback* tells Sayonara to play the next song seamlessly on top of the current one – ideal for playing live albums or classical music. The *cross-fader*, on the other hand, gently fades from one track to the next. The duration of the transition can be adjusted.



**Figure 4:** Streaming music via SomaFM or SoundCloud worked in the test, but only with the Sayonara version from the Fedora package sources.

If you prefer audio books to classic paper books, you might want to check out the options below *View | Speed/pitch*. Here, Sayonara offers you the possibility to change the playback speed. If you also enable the *Hold Pitch* option, the audio player will attempt to keep the original pitch.

### Data Streams

Podcasts don't cause Sayonara any problems either, but the program doesn't have a search engine for them. You therefore have to enter your favorite

shows manually. Open the corresponding module via *View | Podcasts* and select *New* from the hamburger menu. Enter the name and URL of the podcast's RSS stream in the dialog. Sayonara then loads the individual contributions of the podcast in the playlist and displays the podcast's cover.

Sayonara has done a better job of integrating Internet radio. Use *View | Streams* to open the required module. In the hamburger menu, you will find the *Search for radio stations* option – the audio player accesses the database at

*fmstream.org*. In the dialog, type in the desired radio station and then select the specific program and the audio quality from the hits. Click on *Add* to add the station to the program.

Additionally, Sayonara allows direct access to the services of SomaFM [3] and SoundCloud [4]. You can select either using the checkbox to the right of the cover display, which reads *Music* by default. In the test, these options only worked under Fedora 30 (Figure 4). When selecting SomaFM in the PPA version installed under Ubuntu 19.04, the music library only displayed *Loading SomaFM*; it was not possible to play songs from the source. SoundCloud worked, but artists and albums were consistently described as *n.a.* and *None*.

If you want to record streams, you can do this directly from the player. However, you first need to enable the function. To do this, open the *Settings* again and switch to the *Stream Recorder* tab. When you get there, click on the button for *Active* and select a *Target directory*. Sayonara stores the recorded titles there. In the *Session Directory* tab, you can adjust the nomenclature used to name the recorded files.

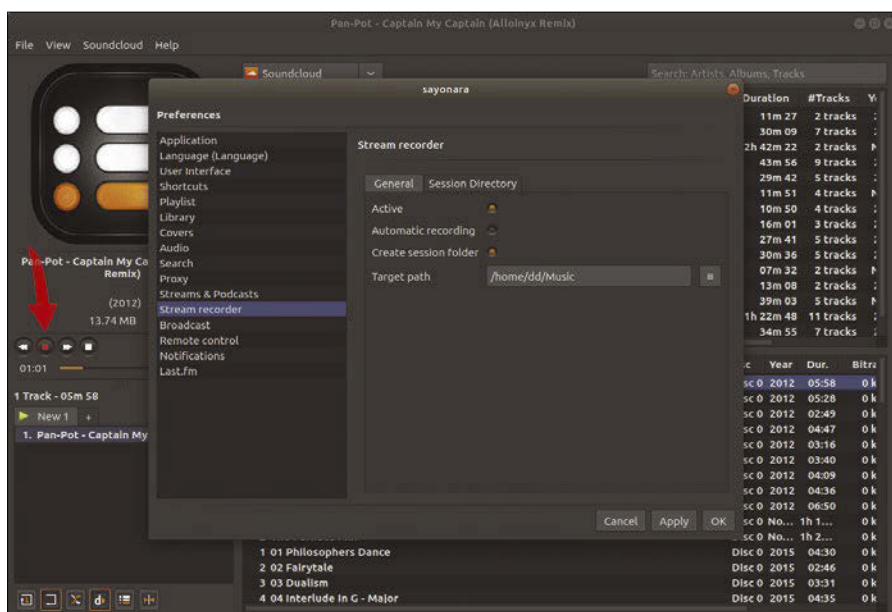
To create a recording, start the desired stream as usual using the *Play* button, which changes the icon to a red square. If you now click on the button again, Sayonara will record the program. You can identify the recording by the “light” on the button (Figure 5). If you press the button again, the audio player stops recording.

### Conclusions

Sayonara is useful as an audio player that not only supports most of the popular formats, but also streaming. Whether it be web radio, podcasts, or services like SoundCloud or SomaFM, Sayonara manages this with very little CPU load and is happy with around 50MB RAM. To cover all the requirements of an audio player, all it would need is a feature for playing audio CDs. ■■■

### Info

- [1] Sayonara Player: <https://sayonara-player.com>
- [2] Sound Juicer: <https://wiki.gnome.org/Apps/SoundJuicer>
- [3] SomaFM: <https://somafm.com>
- [4] SoundCloud: <https://soundcloud.com>



**Figure 5:** The built-in stream recorder lets you record broadcasts from web radio stations or the supported streaming services.



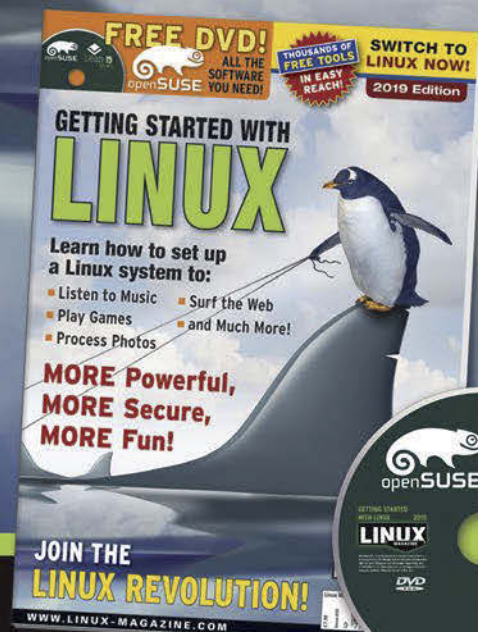
# Harness the power of Linux!

Want your friends and colleagues to make the switch to Linux?

This single issue shows beginners how to:

- install Linux
- download and install free software for your Linux system
- play games
- create documents and spreadsheets
- process photos
- play music and videos
- and much more!

## GETTING STARTED WITH **LINUX**



ORDER ONLINE:  
[shop.linuxnewmedia.com/specials](http://shop.linuxnewmedia.com/specials)



Analyze disk usage with Baobab

# CLEANUP!

Unnecessary files and directories take up valuable space on hard disks or SSDs. Baobab lets you locate and remove data garbage at the push of a button. *By Erik Bärwaldt*

Computer mass storage is getting bigger and cheaper. With the rapidly increasing capacity of hard disks, SSDs, and USB sticks, users are tempted to store even more information (with some of it soon forgotten).

To find your way through the data jungle, Linux offers command-line tools like `du` or `df`. However, these tools do not provide detailed information on where data is stored or which files are the biggest space hogs.

Baobab [1], also known as Disk Usage Analyzer, is a graphical, menu-driven viewer, which lets you quickly track down obsolete data and identify space hogs. Baobab also lets you remove useless files without requiring massive search overhead.

You can find Baobab in the package sources of virtually all popular Linux distributions. It is easy to install using your distribution's package management tools. Some distributions, such as Ubuntu, automatically install Baobab.

Although the tool works with the GTK+ toolkit, Baobab also works with GTK desktops other than Gnome. After installation, the tool will appear in your desktop's menu hierarchy as Disk Usage Analyzer.

## Windows

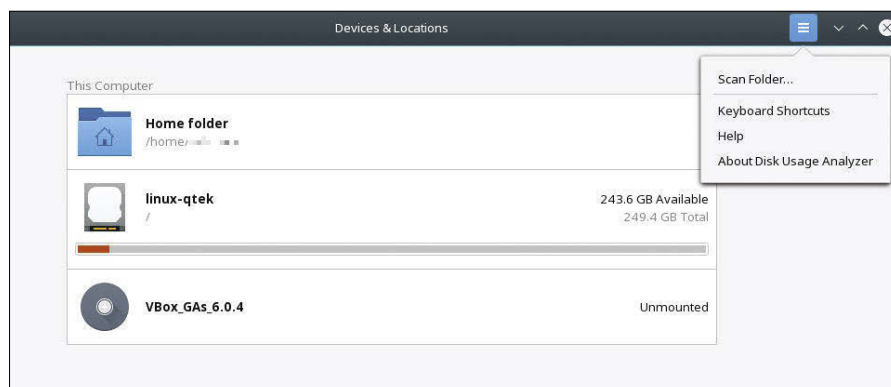
When launched, Baobab opens the Devices & Locations window, which shows all the drives available on your system, regardless of their mount status. Here you can see a drive's size and capacity to the right of the drive designation, as well as a colored bar beneath that shows total drive utilization at a glance (Figure 1).

Clicking on a displayed drive opens a two-paneled window that shows the selected drive's directory hierarchy in a tree structure on the left. Baobab again uses colored bars to indicate utilization, which ranges from red (almost full) to yellow (partially full) to green (almost

empty). If a small triangle appears in front of the directory name, clicking on it expands the view to the next hierarchy level. To the right of the directory tree structure and utilization bar graphs, you will find each level's absolute size and the number of objects it contains.

In the right pane, the drive's space allocation is shown as a multilayer pie chart, with each layer representing a further hierarchy level. If you mouse over the pie chart, Baobab temporarily displays the corresponding subdirectories and the occupied storage capacity (Figure 2).

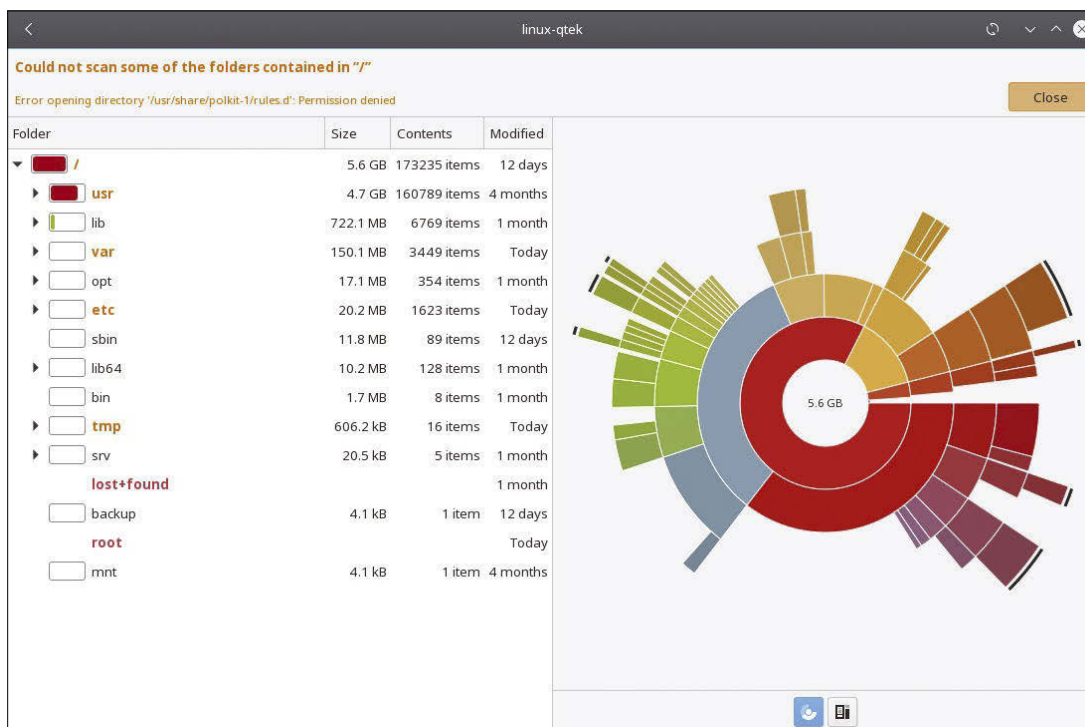
If you click on another directory in the tree view on the left, the view also



**Figure 1:** When launched, Baobab shows you all the drives available on the system with their utilization levels.

Lead image © Amy Walters, 123RF.com





**Figure 2:** Directories with the largest amounts of data can be located quickly.

changes in the right window segment. Using the two buttons at the bottom of the right pane, you can switch from a pie chart (the button on the left) to a tile graph (button on the right). If the view appears too detailed (or too general for complex hierarchies), you can also use the context menu to zoom in or out.

## Administration

Baobab not only provides insight into the data carriers' structure and assign-

ments, but it also supports basic administrative tasks. When you right-click on a desired directory in the right pane, a context menu pops up that allows you to open the current working directory in the desktop's file manager, letting you work with its data. Alternatively, you can move a selected directory (including all its subdirectories) directly from Baobab to the trash can.

The left tree view also provides administrative task functions via a con-

text menu, which you open by right-clicking on the desired directory. However, the only options available from this pane are for saving the file path to the clipboard, moving the directory with its subdirectories to the trash can, or opening the folder.

On computers with large volumes of data stored in deeply nested structures, it makes sense to

work with individual folders. To do this, click on the *Scan Folder...* option (located under the hamburger icon drop-down list in the top-right corner of the drive view) and then select the desired folder in the file manager. After closing the dialog, Baobab displays the folder contents in the usual way, with the selected folder appearing as the root directory in the tree structure.

## Removed

In addition to scanning locally mounted media, Baobab also

supports media analysis and data cleanups on remote computers. To connect to an external computer, select *Other Locations* located bottom left in the Select Folder dialog box (Figure 3).

The file manager now displays other systems found on the local network. You can connect to the remote computer by clicking on its icon; if this does not work, enter the server address in the input field in the bottom right corner.

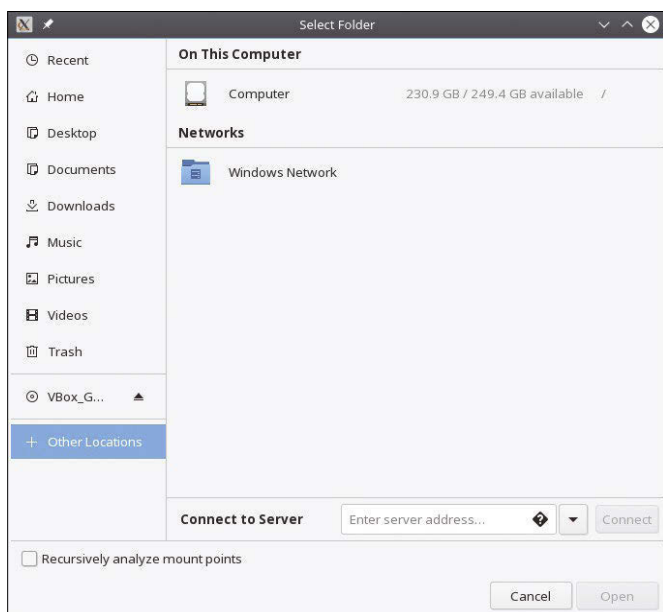
After pressing *Connect*, Baobab opens the corresponding connection. The software then switches back to the main window and displays the contents of the remote computer or server, which can be used just like local files and directories.

## Conclusions

Baobab brings order to chaotic file and directory structures on mass storage media. If you run out of disk space, Baobab helps you quickly find the biggest storage hogs and move them directly to the trash can, bypassing the command line. The tool impresses with its ease of operation, good overview, and option to work on remote systems. ■■■

## Info

[1] Baobab: <http://www.marzocca.net/linux/baobab/>



**Figure 3:** Baobab can also remove obsolete data from remote systems.



## The sys admin's daily grind: ntpviz

## And the Time Is ...

The Network Time Protocol allows admins to keep time on their computers. Due to the way the system works, this timekeeping is only moderately successful. Charly uses the ntpviz statistics tool to visualize time fluctuation. *By Charly Kühnast*

I recently browsed the NTPsec repository [1], a heavily reworked fork of the well-known Network Time Protocol daemon, ntpd. The newcomer

is looking to ditch legacy ballast and finally provide protection against Man-in-the-Middle attacks. NTPsec is not the topic today, but I would like to talk

about a small tool that I found while browsing: ntpviz. Among other things, the program visualizes the extent to which the time queried by the NTP server deviates from the local time (offset) and how strongly it fluctuates (jitter).

To get started, I cloned the Github repository and started the installation:

```
cd /usr/local
git clone --depth 1 https://gitlab.com/NTPsec/ntpsec.git
cd ntpsec
./buildprep
./waf configure --refclock=all &&
./waf build && ./waf install
```

Then I created the directory where the statistics data will be stored:

```
mkdir /var/log/ntpstats
```

What's missing is /etc/ntp.conf from Listing 1. Now I can start the NTP daemon. With the parameter -N it runs with increased priority – this improves the accuracy:

```
ntpd -c /etc/ntp.conf -N
```

Now, I have a day off – after all, ntpd has to collect enough statistics. Typing

```
ntpviz @day/optionfile
```

was supposed to start the visualization, but it blew up in my face the first time I tried it. It turns out that Gnuplot has to be installed – a fact that slipped by me when checking the dependencies. After installing Gnuplot everything runs like clockwork: The www/day subdirectory contains an HTML file with various graphs (Figure 1). ■■■

## Info

[1] NTPsec: <https://gitlab.com/NTPsec/>

## Author

**Charly Kühnast** manages Unix systems in the data center in the Lower Rhine region of Germany. His responsibilities include ensuring the security and availability of firewalls and the DMZ.

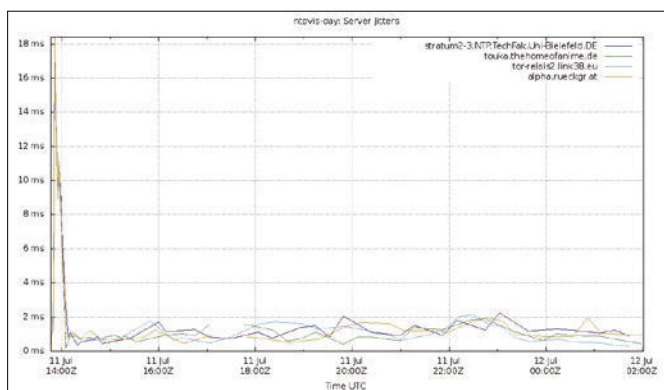


Figure 1: Jitter statistics of four NTP servers queried over a period of 12 hours.

## Listing 1: /etc/ntp.conf

```
01 server 0.de.pool.ntp.org
02 server 1.de.pool.ntp.org
03 server 2.de.pool.ntp.org
04 server 3.de.pool.ntp.org
05
06 restrict -4 default notrap nomodify nopeer noquery
07 restrict -6 default notrap nomodify nopeer noquery
08
09 restrict 127.0.0.1
10 restrict ::1
11
12 driftfile /var/lib/ntp/drift/ntp.drift # path for drift file
13
14 logfile /var/log/ntp.log # alternate log file
15
16 statsdir /var/log/ntpstats # directory for statistics files
17 filegen peerstats file peerstats type day enable
18 filegen loopstats file loopstats type day enable
19 filegen clockstats file clockstats type day enable
20
21 keys /etc/ntp.keys # path for keys file
22 trustedkey 1 # define trusted keys
23 requestkey 1 # key (7) for accessing server variables
24 controlkey 1 # key (6) for accessing server variables
```



What?!  
I can get my  
issues  
SOONER?



Available anywhere, anytime!

Sign up for a digital subscription and enjoy the latest articles on trending topics, reviews, cool projects and more...

[shop.linuxnewmedia.com/digisub](http://shop.linuxnewmedia.com/digisub)





## Adding dialog boxes to shell scripts

# Let's Dialog!

Create dialog boxes with checkboxes, progress bars, and many other features that users may find helpful when working at the command line. *By Bruce Byfield*

**M**any Bash scripts do not need interfaces. They are run by admins, who are comfortable at the command line. However, if a script is used by everyday users, it may be more friendly if it uses a dialog box for input and messages. For over 25 years, a leading command for boxes has been `dialog` [1], which can be called from a script so that users can enter input in an ncurses interface. It then returns them to the script when they exit.

The command structure of `dialog` is somewhat unusual (Listing 1). In addition to the appearance of options in two separate places, note the quotation marks around the text, and the order of height, width, and other box type options. Height is expressed as the number of lines and width as the number of monospaced characters.

The design of the resulting box varies with the options selected. However, it

will always have text and may have buttons (such as *Yes*, *No*, *Help*, or *OK*) or radio buttons for making selections. It may also have a back title for easy identification. When necessary, the box can be navigated by using arrow keys, and it is not supported by a mouse (Figure 1). Once a selection is made, then generally the script will continue with if/then/else statements that correspond to each selection.

Generally, the easiest way to design a box is to open another prompt and run the basic `dialog` command repeatedly until you have a structure ready to add to the script. Use the Esc key to return to the prompt after the resulting box is displayed. Unless the background for the dialog box matches the

background color for the terminal, you will see conflicting colors after returning to the prompt unless you enter the clear command or a sufficient number of new commands.

### Using Box Options

It makes sense to begin with the available types of boxes, one of which must always be used in a dialog command. All boxes can use the self-explanatory text, height, and width options. Text appears after the box type in quotation marks, while height and width options follow, expressed in lines and characters respectively.

Other options must follow the width. Available options depend on the type of box, as shown in Table 1.

### Listing 1: Command Structure for dialog

```
dialog --"COMMON-OPTIONS" --BOXTYPE "TEXT" HEIGHT WIDTH --BOXTYPE-OPTIONS
```

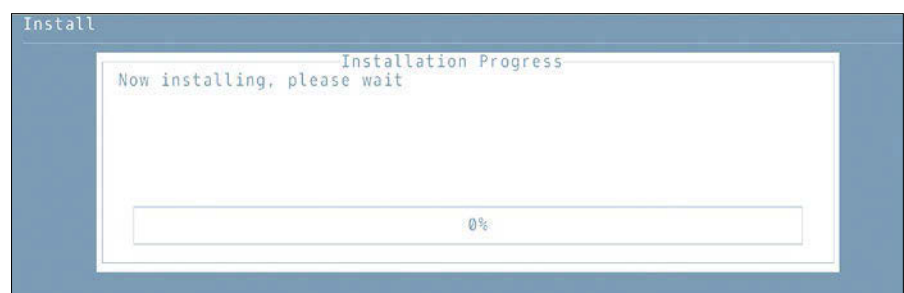


Figure 1: A progress bar with a back title in the top left corner.

### Author

**Bruce Byfield** is a computer journalist and a freelance writer and editor specializing in free and open source software. In addition to his writing projects, he also teaches live and e-learning courses. In his spare time, Bruce writes about Northwest coast art (<http://brucebyfield.wordpress.com>). He is also cofounder of Prentice Pieces, a blog about writing and fantasy at <https://prenticepieces.com/>.



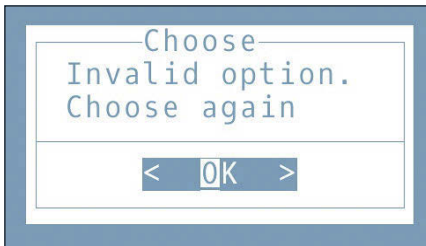


Figure 2: A simple message box.

### Listing 2: Checklist

```
dialog --help-button --checklist
"Choose one of the following:" 10 40 3 \
  1 Accept on \
  2 Reject off \
  3 Modify off
```

A simple message box (Figure 2) would use a command like:

```
dialog --title "Choose" --msgbox 2
'Invalid option. Choose again' 6 20
```

A more complicated example is a checklist, which asks for a selection (see Listing 2 and Figure 3).

Listing 2 shows items prefaced with an identifying tag and followed by a status for the default display, as well as the end of line markers.

Other boxes have similar structures, modified by any common command options. For example, the message box given above might have a help button added to it with the addition of a `--help` button option (Figure 4).

For a complete description of each option, see the man page. Some options have restrictions, most of which are logical enough when you stop to think. For example, a help button can only be added to checklist, radio list, and menu boxes. There would be no point to adding a help button to a message box, for instance, which requires no user interaction.

Screenshots of other boxes are available online [2].

## Common Dialog Options

Common dialog options affect the look and general functionality of the command. `--ascii-line` replaces the ncurses widgets used to create a box with plus and minus signs instead (Figure 5). Ordinarily, a box is centered on the screen, but you can also use `--begin Y X` to set the vertical and horizontal coordinates

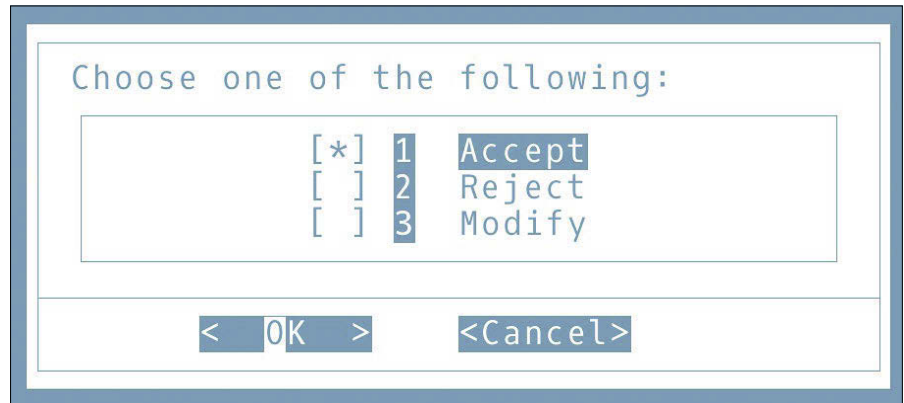


Figure 3: A checklist box, with the default choice set to *Accept*.

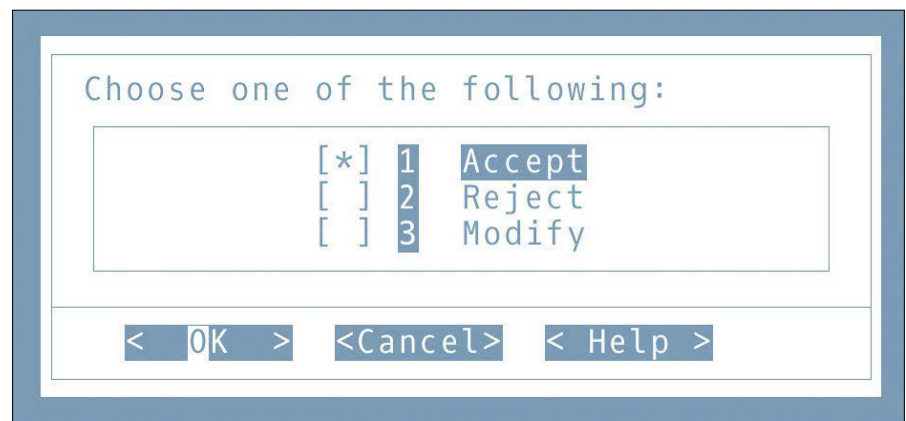


Figure 4: The same checklist box as in Figure 3, but with the addition of a *Help* button.

for the upper left corner of the box. You can also set the dimensions of a box with `--aspect WIDTH/HEIGHT`. With `--scrollbar`, a scrollbar is added to the box, while `--no-shadow` suppresses the default shadow to the right and bottom of the box – an option which gives ncurses a flatter but more modern-looking appearance.

`--color` expresses settings prefaced by a `\Z`. It uses the numbers 0-7 to choose ANSI colors: black, red, green,

yellow, blue, magenta, cyan, and white – in that order. However, contrary to what you might expect from the name, `--color` also sets font weights and effects: `b` sets bold, and `B` turns off bold, while `u` turns on underlining and `U` turns it off. The settings are cumulative, so `Z\u3` produces green, underlined text. To restore default settings, use `\Zn`.

Many of the common options have to do with how the buttons in a box are used. Options like `--no-cancel` suppress

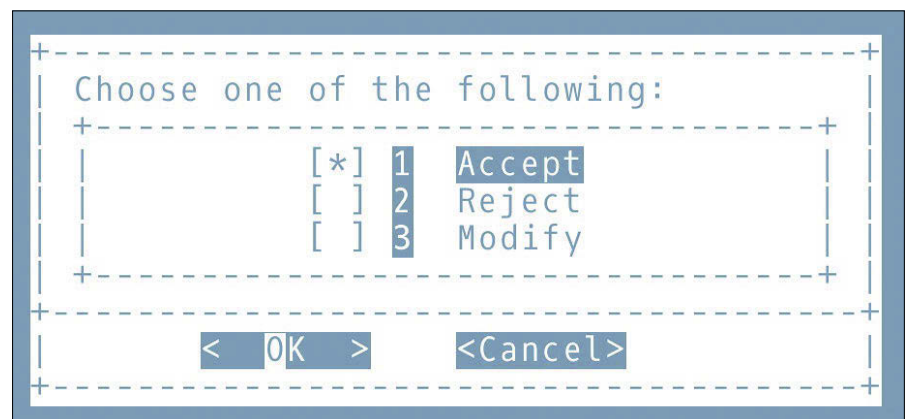


Figure 5: The same box as in Figure 4, but with boxes drawn with ASCII line art rather than ncurses widgets.

**Table 1: Selected Boxes and Their Options**

<code>--background</code>	Like <code>tail</code> , but runs in the background	
<code>--calendar</code>	Displays calendar	<code>--day --month --year</code> : For current date
<code>--checklist</code>	A scrolling list from which to select multiple items	<code>--list-height</code> : Tag item status format for default choices (e.g., <code>1 Accept on \</code> )
<code>--dselect</code>	Selects a directory from a list	<code>--filepath</code>
<code>--editbox</code>	Allows editing of an existing file	<code>--filepath</code>
<code>--gauge</code>	A progress bar	<code>--percent</code> : Displays completion percentage
<code>--info</code>	A message display	
<code>--input</code>	A text entry field for answering questions	
<code>--menu</code>	A scrolling list from which to select one item	<code>--menuheight</code> : Tag item status format for default choices (e.g., <code>1 Accept on \</code> )
<code>--message</code>	OK button	
<code>--passwordbox</code>	For password entry	
<code>--pause</code>	Shows meter for time paused	<code>--seconds</code>
<code>--radiolist</code>	Shows radio boxes for selection	Tag item status format for default choices ( <code>1 Accept on \</code> )
<code>--tail</code>	An auto-updating viewer for the end of files	
<code>--timebox</code>	Selects time	<code>--hour --minute --seconds</code>
<code>--text</code>	A scrollable text box	
<code>--yes/no</code>	Yes and No answer buttons	

All boxes can use `text`, `height`, and `width`. Options can be completed to `dimension`, `time`, `filename`, or some other variable as needed.

the display of a specific button altogether. Others, like `--help` button add a button to types of boxes that could use one; a gauge box, for example, would have no use for a Help button, since there is no interaction. Others, like `--exit-label STRING`, `--ok-label STRING`, `--no-label STRING`, and `--yes-label STRING` replace the standard button titles, making the buttons more versatile. In boxes like menus, checklists, and radio lists that require a selection of a list of items, you can also use `--item-help` to add help for each item, and `--default-item STRING` to select which choices are selected by default.

Still other items affect what happens after a dialog displays. For example, `--stdout` prints the result of a dialog interaction to standard input and `--stderr` to standard error, which may simplify scripting. Script writers may also use `--sleep SECONDS` to pause a script after it processes input from a dialog box or `--and-widget` to force dialog to move to the next box after the current one is processed without resuming the script.

`dialog`'s options total in the dozens, and setting up the command structure can be laborious. For this reason, you may want to run `dialog --create-rc`

`FILE` in your home directory to create a default look and function for the command. The file uses the current settings of your current terminal. Although you probably want to leave the top of the line untouched, there are many choices that are self-explanatory enough to edit. This file can be overridden by options entered at the command

line, but can save considerable time if you use `dialog` regularly (Figure 6).

## Alternatives to Dialog

While `dialog` is the most common option for adding partial interfaces to scripts, both `whiptail` [3] and `zenity` [4] offer a similar, if less complete command structure. However, `zenity`'s man page is perhaps needlessly complex, dividing options into 10 different categories that at first make it difficult to see the overall command structure. Moreover, `zenity` uses `GTK+` rather than `ncurses`.

In addition, `Xdialog` [5] is intended as a drop-in replacement for `dialog`, and is referenced in `dialog`'s man pages to mention a few minor differences. However, `dialog` remains the most extensive and more commonly used. As primitive as `ncurses` may appear to modern users, it is generally adequate for the purposes of most scripts. ■■■

## Info

- [1] `dialog`: <https://invisible-island.net/dialog/>
- [2] More dialog screenshots: <https://linuxgazette.net/101/sunil.html>
- [3] `whiptail`: [https://en.wikibooks.org/wiki/Bash\\_Shell\\_Scripting/Whiptail](https://en.wikibooks.org/wiki/Bash_Shell_Scripting/Whiptail)
- [4] `zenity`: <https://wiki.gnome.org/action/show/Projects/Zenity>
- [5] `Xdialog`: <https://linux.die.net/man/1/xdialog>

```
# Set tab-length (for textbox tab-conversion).
tab_len = 0

# Make tab-traversal for checklist, etc., include the list.
visit_items = OFF

# Shadow dialog boxes? This also turns on color.
use_shadow = OFF

# Turn color support ON or OFF
use_colors = OFF

# Screen color
screen_color = (CYAN,BLUE,ON)

# Shadow color
shadow_color = (BLACK,BLACK,ON)

# Dialog box color
dialog_color = (BLACK,WHITE,OFF)

# Dialog box title color
```

**Figure 6: Part of the `dialog` configuration file.**

# IT Highlights at a Glance

The collage features three newsletter preview images. The top one is for ADMIN HPC, the middle for ADMIN Update, and the bottom for Linux Update. Each preview shows various articles, highlights, and promotional banners for the respective newsletters.

Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your Inbox. Subscribe today for our excellent newsletters:

ADMIN HPC • ADMIN Update • Linux Update  
and keep your finger on the pulse of the IT industry.

Admin and HPC: <https://bit.ly/HPC-ADMIN-Update>  
Linux Update: <https://bit.ly/Linux-Update>





Online password protection

# Password: password

Securely storing passwords online can be a complex task. With a few tools, websites can offer better security, but users still need to choose their passwords wisely. *By Stefan Wintermeyer*

**W**hen it comes to password security on social networking sites, such as LinkedIn or XING, users usually have to rely on the website’s claims. Because these sites are closed source applications, users don’t know for sure how their passwords are protected. Often, by the time they find out, it may be too late. In 2012, for example, LinkedIn lost an estimated 6.5 million encrypted passwords. There may have

been more as the company did not provide exact figures, because in 2016 a hacker offered to sell 117 million LinkedIn users’ customer data [1].

Of all the types of stored data, user passwords are a particularly attractive target. One reason for this is that many users reuse a password for more than one site. Once an attacker has cracked a user’s password on one site, they can use it to exploit other sites as well. Another reason is that the passwords many people choose are just too simple.

While all social networks store personal data, open source solutions, such as my project vutuv [2], are usually more open with their security approaches (see the box “Open Source Infrastructure”). Read on for a behind the scenes look at how our site goes about securely storing passwords.

ated Magic Link that would enable logging in without a password.

Quite proud of our secure Magic Link solution, we awaited community praise after the platform launch. Instead, we started getting support requests: Where do I enter a password? How do I set the password? The answer that vutuv did not need a password was something that none of the users found worthy of compliment.

We learned from this experience. For end users, it’s completely normal to log on to a website with some kind of account name and password and to reset forgotten passwords by email if necessary. For 99 percent of our users, logging on with a Magic Link turned out to be non-intuitive and did not meet our target audience’s needs. At the beginning of 2019, we decided to develop a second version, which among other things required a legacy login password [2].

### Open Source Infrastructure

Vutuv runs its servers on Debian Linux (stable), which automatically installs security patches overnight. For the firewall, we use Shorewall [3]. While version 1 of vutuv’s web application relied on MariaDB [4] as its database, from version 2 onward vutuv uses PostgreSQL [5]. NGINX [6] is the web server component. The application itself is based on the Phoenix Framework [7], which lets you roll out new versions via hot deployment without downtime, and is written in the Elixir [8] programming language.

### Password-Free Is Passé

In vutuv’s first version, we completely dispensed with user passwords and used a Magic Link instead. When logging in, a user only had to enter their email address; they were then sent a newly cre-

### Second Try

Instead of reinventing the wheel for password functionality, we implemented the Phoenix authentication library, Phauxth [9]. Since we had special system require-

Lead Image © Kian Hwi Lim, 123RF.com

ments, we quickly entered into an exchange with the library inventor, David Whitlock, who now is working full time for the vutuv core team on version 2.0.

For any web application, the worst case scenario is a hijacked server. An attacker would then have full access to the database and all the configuration files on the server. In vutuv's case, the attacker would probably have grabbed the source code from our public GitHub repository [2] in advance and studied it. For that reason, we do not use an encrypted database. Not only does it impact speed, but if an attacker has complete access to your system, encryption is pointless. The Phauxth authentication library ensures that an attacker who gets on your server system will not be able to extract passwords.

## Brutally Simple

The simplest attack is a dictionary attack in which attackers try out popular passwords using brute force. This type of attack doesn't even require breaking into the target system. However, the attacker does need an account name that uses a weak password.

To prevent brute force attacks of this kind, vutuv uses a login limit. After

three unsuccessful attempts, a user has to wait a few minutes until the next attempt. After further unsuccessful attempts, the wait increases and the system triggers an internal alarm.

## Hashes

Hopefully, no one saves passwords in plain text, as this would give an attacker access to all the stored passwords. Instead, the application generates a password hash and saves it. One type of password hash uses the MD5 message-digest algorithm. If a user enters the password *banana*, the application will generate the MD5 hash `72b302bf297a228a75730123efef7c41` and store the hash in its database. When the user logs in again with *banana*, the program passes on the password's hash value to the database for comparison. If an attacker gained access to this database, the assumption is that the attacker would have a whole bunch of unusable information, because the password cannot be decoded from the hash.

Today, MD5 hashes are considered insecure, because resourceful hackers have found a way to decipher the hashes. Using brute force to create a table with the hashes of all possible

password combinations, hackers developed a rainbow table making it easy to decode the hash. In the MD5 example using `72b302bf297a228a75730123efef7c41`, you can now simply google the string, and the search engine will come up with links to ready-to-use rainbow tables with the solution (Figure 1).

## Adding a Pinch of Salt

To remedy this, you can extend the original password with an additional password that only the server knows – in other words, a salt. If you store a salt with a value of `fasiurw24089sda` on the server and add the insecure user password *banana* to it, the result is a secure password that does not occur in any existing rainbow table.

While this sounds like a good solution, site operators must assume that an attacker also has access to this salt. In addition to stealing the database dump, the attacker could also have a copy of the complete configuration, which means that the salt is known. Since MD5 no longer generates any serious CPU load, the attacker could create a new rainbow table with this salt. And if money isn't a consideration for the attacker, they could use an Amazon Web Services cluster to do this in a short time.

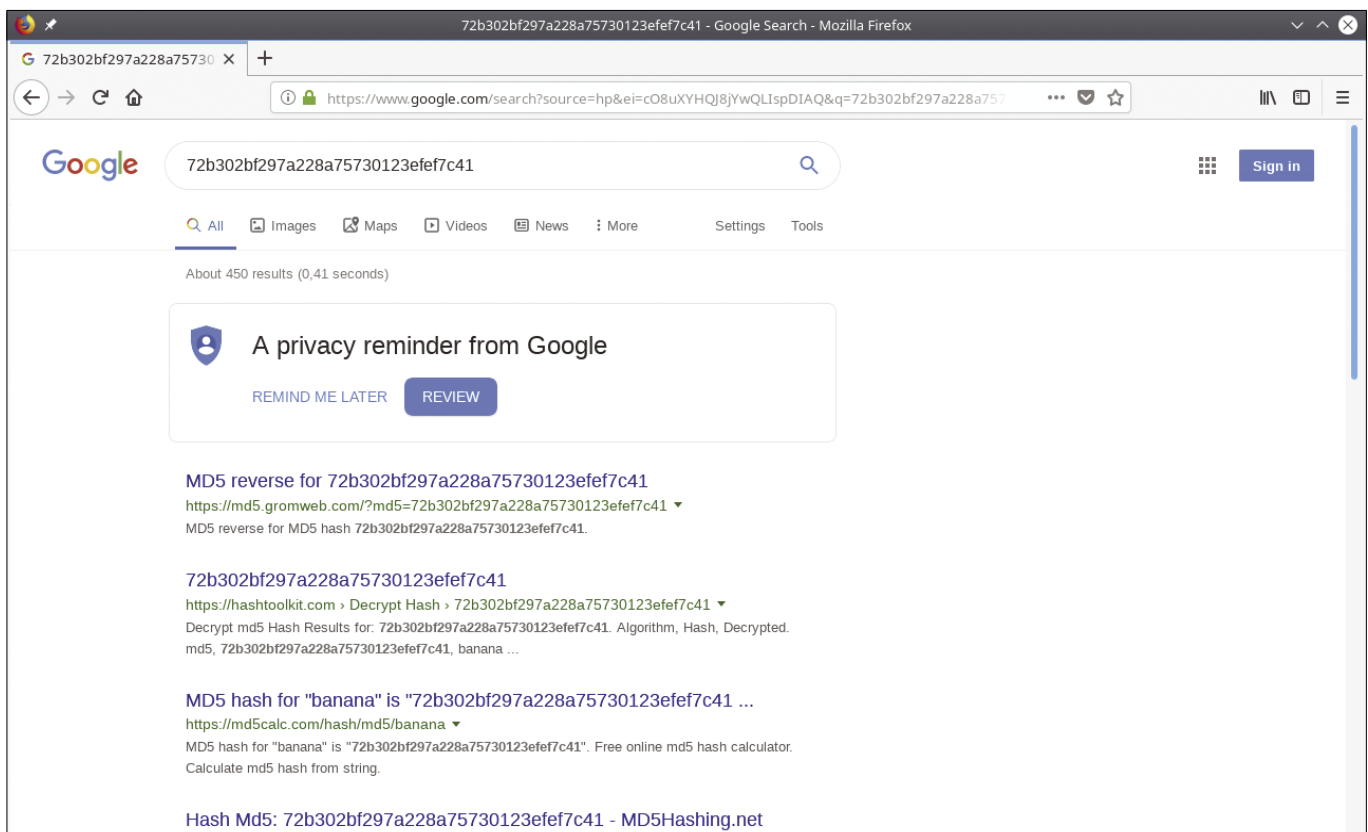


Figure 1: MD5 hashes can easily be translated back to the original passwords.

```

- 123456
- password
- 123456789
- 12345678
- 12345
- 111111
- 1234567
- sunshine
- qwerty
- iloveyou

```

**Figure 2: The 10 most commonly used passwords.**

Consequently, a single salt for the entire application is no longer considered secure. The next step is to generate a random salt for each individual account and store it in the database in addition to the hash, which raises the barrier considerably. An attacker would have to calculate the complete rainbow table for each account. However, with the MD5 algorithm and a high budget, this hurdle could be overcome.

Creating complete rainbow tables is only fast and cost-effective if the cryptographic hash function requires very little in terms of hardware resources (CPU and memory). With MD5, the computational overhead is ridiculously low from today's standpoint, which is why brute force attacks go through all possible password combinations.

Therefore, the goal is to create a hash that takes as many computer resources as possible to generate the solution without, of course, inadvertently reaching the other extreme – sluggish response times. After all, it makes no sense for users at login to have to wait a minute for the server to generate the entered password's hash and compare it with the database.

In the last Password Hashing Competition (PHC) in 2015 [10], developers

## Social Engineering

With most attacks, it doesn't matter how securely the website admins store the password if the end user can easily be persuaded by social engineering to reveal the password and – if necessary – the 2FA key. Then a simple phone call and an untrained employee at the target enterprise are all it takes for the attack to succeed.

compared 24 different hashing algorithms. The winner was Argon2 [11], which was developed by Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich of the University of Luxembourg. With Argon2, both the CPU and RAM load can be defined; this prompted vutuv to switch to Argon2, which uses a random 16-byte salt, starting with version 2 and later. Using Argon2 ensures that attackers no longer can create complete rainbow tables.

## Password Choice

However, protecting password databases with hashes is only half the battle. Unfortunately, Internet users tend to use very simple and often common passwords. Figure 2 shows the 10 most commonly used passwords worldwide in 2018.

If an attacker gets a list of the 1,000 or even 10,000 most popular passwords and uses them to create a minimal rainbow table, he can't crack all the passwords, but he can crack many of them.

To protect vutuv users against this attack, we rely on the <https://haveibeenpwned.com> database for version 2. It contains the hashes of 551,509,767 accounts that have already been cracked elsewhere. Anyone using a password that has already been cracked will receive a warning from our login service.

This ensures that a normal system user, who does not care about password security, does not use a password that is really easy to crack. However, we leave it to the end user's discretion to heed or ignore this warning.

However, increasing computer speeds still pose a big risk. Even if an attacker does not create a complete rainbow table today, they may be able to do so in the future. For this reason, I would advise every Internet user to use a password manager that generates a new, random, and unique password for each website.

Internet users should assume that it is only a matter of time before any given account is hacked. In that instance, they will want to make sure that the hacked password does not work on any other site. In practical terms, this only works if users rely on a password manager.

## 2FA

If you want a state-of-the-art solution to password security, you can use two-

factor authentication (2FA). The web application asks for a one-time key in addition to the username and password. In most cases, these one-time keys are generated by mobile phone applications such as Google Authenticator or by special hardware such as YubiKey [12]. Users can also print out a list of one-time keys and cross them out manually with a pen after use. Having said this, texting a one-time key via the server is no longer considered sufficiently secure.

## Conclusion

When it comes to website passwords, guaranteeing security is a complex task for site admins. Hashes, salts, and 2FA are all tools that can help keep users' credentials safe. However, safety is a two-way street. It is still up to the user to provide unique passwords and keep those passwords private (see the box "Social Engineering"). ■■■

## Info

- [1] LinkedIn hacks: <https://www.theguardian.com/technology/2016/may/18/hacker-advertises-details-of-117-million-linked-in-users-on-darknet>
- [2] vutuv: <https://www.vutuv.de>
- [3] Shorewall: <http://www.shorewall.org>
- [4] MariaDB: <https://mariadb.com>
- [5] PostgreSQL: <https://www.postgresql.org>
- [6] NGINX: <https://www.nginx.com>
- [7] Phoenix Framework: <https://phoenixframework.org>
- [8] "Functional Programming with Elixir," by Andreas Möller, *Linux Magazine*, issue 181, December 2015, [http://www.linux-magazine.com/Issues/2015/181/Elixir-1.0/\(language\)/eng-US](http://www.linux-magazine.com/Issues/2015/181/Elixir-1.0/(language)/eng-US)
- [9] Phauxth: <https://github.com/riverrun/phauxth>
- [10] PHC: <https://password-hashing.net>
- [11] Argon2: <https://www.ietf.org/archive/id/draft-irtf-cfrg-argon2-03.txt>
- [12] YubiKey: <https://www.yubico.com>

## Author

**Stefan Wintermeyer** is a consultant, trainer, and book author on the subjects of Ruby on Rails, Phoenix, and web performance.





# REAL SOLUTIONS *for* REAL NETWORKS

ADMIN – your source for technical solutions to real-world problems.

Learn the latest techniques for better:

- network security
- system management
- troubleshooting
- performance tuning
- virtualization
- cloud computing

on Windows, Linux, and popular varieties of Unix.

**GET IT FAST**  
with a digital subscription!

**6 issues per year!**

**ORDER NOW**

[shop.linuxnewmedia.com](http://shop.linuxnewmedia.com)





## Mandatory Access Control with AppArmor

# Armor On

Today's security environment is a tumultuous landscape riddled with threats. AppArmor offers an extra ring of protection for your system, and it is easier to learn and implement than many alternative mandatory access control solutions. *By Shashwat Pant*

**M**andatory Access Control (MAC) is a policy-based security framework that augments conventional security systems by providing an additional layer of protection. Unlike normal permissions, which revolve around performing functions on a filesystem, MAC deals with applications rather than files and directories. MAC doesn't replace existing permissions but supplements them. Since Linux Kernel 2.6, all the MAC versions are implemented over the Linux Security Module (LSM) framework. This article will focus on AppArmor [1], which has been around since 1998 and has been supported by Canonical since 2009.

### Author

**Shashwat Pant** is a freelance writer, coder, and analyst. He loves to read about technology and geopolitics.

One advantage of a policy-based model is that a policy cannot be changed by the user, unlike normal permissions, which can be changed if you have sufficient rights. Another added benefit is that it is applicable to all users, even those with superuser privileges.

### AppArmor: A Beginner's Delight

Policy-based systems have a higher learning curve, but even the most basic MAC will bolster the system security. Many policy-based systems are so complex that users don't delve deeply enough to implement them effectively. One benefit of AppArmor is that it is comprehensive, yet simple enough to deploy with minimal investment of time and training.

Originally created by Immunix in 1998 as SubDomain, the program saw a tu-

multuous development period with an acquisition from Novel in 2005, after which it was renamed AppArmor, before winding up with Canonical. AppArmor has been included in the default Ubuntu configuration since Ubuntu 7.10 Gusty Gibbon.

AppArmor ensures apps behave in a secure manner and doesn't allow functions that the application is not supposed to perform. It takes control of the system resources and acts as a distributor of those resources to the applications – similar to the walled runtime environment.

Two important models for security systems are:

- **Misuse Prevention** – A model that works on blacklisting known harmful entities.
- **Anomaly Prevention** – A model that works on whitelisting entities; it only allows execution of items that are whitelisted and discards everything else.

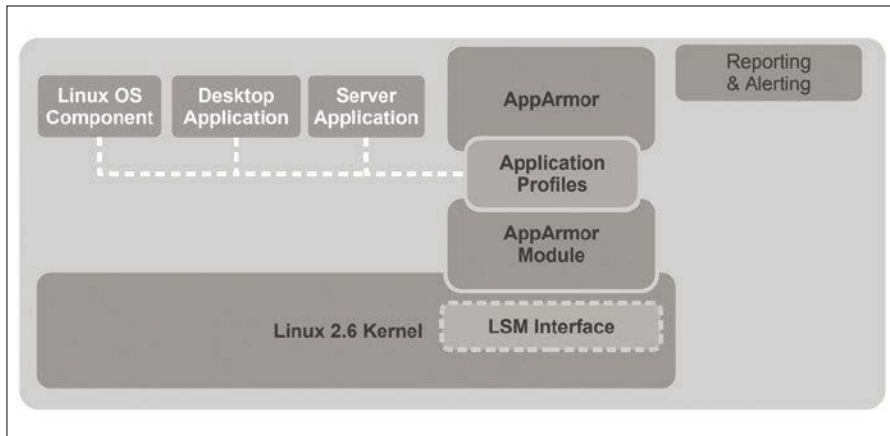


Figure 1: AppArmor architecture.

AppArmor is an amalgamation of both these approaches. For applications that have predefined policies, it acts as a whitelist. For everything else, it is a blacklist and doesn't define a whitelist, although a user can create a policy and define the whitelist for a specific application. AppArmor is application-specific and not user-specific (unlike the alternative Linux MAC tool SELinux).

AppArmor (Figure 1) is implemented as a Linux Security Module (LSM). LSM, which was introduced with Kernel 2.6, is a feature that allows anyone to create a Mandatory Access Control (MAC) policy and insert it in the kernel without the need to wait for a new kernel refresh or maintainer permission.

LSM provides a complete ring of protection, unlike other options that are implemented at the library layer.

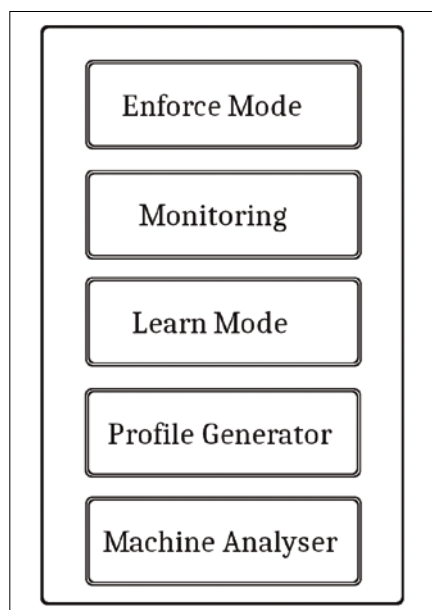


Figure 2: Shows the processes involved in AppArmor.

## Getting Started

AppArmor comes pre-installed on many Debian/Ubuntu- or SUSE-based distros, and packages are also available. See your own distro's documentation for more on obtaining and installing AppArmor. In addition to the base package, be sure you install the `apparmor-utils` and `apparmor-profiles` package for extra tools and profiles.

AppArmor consists of several layered components (Figure 2). Some of the important features are:

- **Machine Analyzer** – Auto-scans ports, discovers applications listening to the port, detects applications without any profiles, and identifies applications that need to be confined with AppArmor.
- **Profile Generator** – Static analyzer that identifies the program either as an x86 application or a script and provides a basic template.
- **Learn Mode** – In this mode, the rules are not enforced but the program is put under a listen mode and any violations are logged.
- **Optimizer** – Learns from the violations logged and provides iterative questions that help to create a thorough profile with user consent.
- **Enforce Mode** – In the mode, the new profile is put under check and complete enforcement is applied; all violations are stopped.

Once you have completed the setup, use the `aa-enabled` command to check whether the module is functioning. To ascertain what extra functionality it comes with, run the `aa-status` command with `su/sudo` and check the output. You will notice an output showing profiles being loaded in different modes.

Each profile is monitored under two modes:

- **Complain or Learning or Unconfined Mode** – In this mode, the policies are not enforced and the app can work without any restrictions. However, the policies are monitored and all actions will be traced in the logfile. Programs under this profile are governed by normal permissions.
- **Enforce Mode or Confined Mode** – Policies in this mode are strictly enforced and any violations will be stopped. All actions are traced and inferred to the logfile.

Profiles are written from the perspective of processes and not the system. Creating and managing profiles is the critical task for using AppArmor effectively, and the overall security hinges upon the privileges granted to each profile.

All the profiles created are stored in `/etc/apparmor.d` directory and are named after the location of the file – with the `/` separator in the filename replaced with a dot (`.`). For instance, a script file named `test` stored in `/home/linuxmagazine` will be named `.home.linuxmagazine.test`.

There are multiple ways to create a profile, and the process depends upon the granularity with which you want to filter and monitor the application. You can use the simple profile tool `aa-autodep` or the more exhaustive utility `aa-genprof`.

`aa-autodep` is a basic tool that automatically scans and forms a basic profile of an app or a script. The profile thus created is in complain mode, and as a result, the rules defined are not enforced and only the actions are logged.

### Listing 1: Test Script

```
01 #!/bin/bash
02 #AppArmor test script
03
04 touch test.txt
05 echo "File created"
06
07 cp /home/shashwat/hello.text .
08 echo "File Copied"
09
10 rm test.txt
11 rm hello.text
12 echo "File has been deleted"
13 echo "Test successful"
```



aa-genprof offers a more thorough scan of the app or script and generates a complete AppArmor profile. The profile thus created is put in enforce mode, and full compliance of the policy is ensured. Any violation of the policy will result in termination of the process, and changes will be logged.

### Scanning an Application

Before I get to the gritty details of profiling, I need a small script. I will use the program shown in Listing 1.

The test script in Listing 1 performs basic functions of file creation, copying, and removing the files. Four utilities are used in the test script: touch, cp, rm, and echo. All these utilities require different permissions to operate, and this is where AppArmor strikes a chord; it will scan the app recursively, monitoring every aspect of the script.

Save the test script in any folder and rename it aatest. Be sure to give it the execute permission with:

```
chmod u+x aatest
```

I will start by generating a new profile with the following command:

```
aa-genprof aatest
```

The utility will ask you to run the script in a different terminal to start the scan process. Execute aatest in another window, and once it is completed, return to the scan window. Press s to start the scan. The scanning process consists of two steps:

- utility access
- action(s) performed

Utility access allows you to borrow the right to access a tool. The output from this mode (Listing 2) shows three facets of the app: Profile, which displays the absolute address of the script/app that you are scanning (aatest in this case); Execute, a utility that the script is asking to run (touch, in this case); and Severity, which shows the danger level of the utility (from 1-10 with 1 being lowest security and 10 being highest). The severity will appear for each utility being used in the script (in this case, touch, cp, and rm).

The second step deals with the action performed by the utilities. The user can decide how to address each action per-

formed by the script. Options include:

- Inherit (ix) – Inherits the property of the binary file.
- Child (Cx) – Creates a sub-profile within the main profile; the inherited binary will also be monitored under complain mode.
- Deny – The resource will be denied during the execution of the program.
- Abort – Exit the AppArmor profile scan without writing any changes to the profile.

- Finish – Exit the profile scan after writing the changes to the profile. Because I will use the test script in the terminal, rights to read and write in teletypewriter (tty) will also be included, in addition to the actions performed by other tools, such as write, read, and execute functions performed in the scripts. This step also includes a modified parameter for file functions. In Listing 3, you can see a new path variable that shows the path of the output, since the

#### Listing 2: Scan Output

```
01 [(S)can system log for AppArmor events] / (F)inish
02 Reading log entries from /var/log/syslog.
03
04 Profile: /home/shashwat/appar_t/aatest
05 Execute: /bin/touch
06 Severity: unknown
07 (I)nherit / (C)hild / (N)amed / (X) ix On / (D)eny / Abo(r)t / (F)inish
```

#### Listing 3: New Mode Section

```
01 Adding #include <abstractions/consoles> to profile.
02 Profile: /home/shashwat/appar_t/aatest
03 Path: /home/shashwat/appar_t/test.txt
04 New Mode: owner w
05 Severity: 6
06
07 [1 - owner /home/*/appar_t/test.txt w,]
08 2 - owner /home/shashwat/appar_t/test.txt w,
09 (A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew /
Audi(t) / (O)wner permissions off / Abo(r)t / (F)inish
```

#### Listing 4: Permission Denied

```
01 shashwat@shash-nix:~/appar_t$ ./aatest
02 This is a test for apparmor capabilities.
03 ./aatest: line 6: /bin/lm: Permission denied
04 File created
05 File Copied
06 File has been deleted
07 Test successful
```

#### Listing 5: aa-logprof Command

```
01 sudo aa-logprof
02 [sudo] password for shashwat:
03 Reading log entries from /var/log/syslog.
04 Updating AppArmor profiles in /etc/apparmor.d.
05
06 Profile: /home/shashwat/appar_t/aatest
07 Execute: /bin/lm
08 Severity: unknown
09
10 (I)nherit / (C)hild / (N)amed / (X) ix On / (D)eny / Abo(r)t / (F)inish
```

first function of the test script deals with the creation of a new file `test.txt` in the same directory. There is also a new mode section that lists the permission required to perform the function.

Once you have finished granting access to the binaries and function, finish the scan process by pressing `f`; after this, your program will be monitored within AppArmor.

## Monitoring Changes

The process described in the preceding section works well for a constant script. If any changes are made to the script or any program, AppArmor will restrict the functioning of the modified code, since those changes were not defined in the profile.

The test script creates, copies, then removes a file. I will now modify the script by adding a soft link to the previously created `test.txt` file. If you run the program again, you will notice that the `ln -s test.txt` command has been denied permission (Listing 4).

To allow this new function, you have to modify the previously created AppArmor profile. Simply, use the command `aa-logprof`, and a new Utility access prompt will appear, asking you to grant new additional privileges to the profile (Listing 5).

Once you grant the additional privileges, if you run the program again, you will not face any issues.

## Predefined Profiles

To explore AppArmor on an app level, I will use a simple example of a generic app with a pre-configured profile

### Listing 6: NTP Profile

```
01 capability ipc_lock,
02 capability net_bind_service,
03 capability setgid,
04 capability setuid,
05 capability sys_chroot,
06 capability sys_resource,
07 capability sys_time,
08 capability sys_nice,
09
10 /var/lib/ntp/*drift rw,
11 /var/lib/ntp/*drift.TEMP rw,
12 /var/log/ntp w,
13 /var/log/ntp.log w,
14 /var/log/ntpd w,
```

and check how insufficient security on a generic app can cause a big vulnerability. Ubuntu used to default to `ntpd` for syncing time. Newer releases have shifted to `timesyncd`, but for more precise time changes, NTP is still the preferred service. The profile for NTP doesn't come bundled with AppArmor profiles, but it comes with the NTP package itself.

NTP requires root access with write privileges on root files, along with the need for an open port to get data from an online server. The combination of root privileges and open port access is a recipe for disaster, and in case of a bug, it can provide an opportunity for a system-wide exploit.

AppArmor curtails NTP's capabilities to limit POSIX permission and file access. The profile limits the capability to write access to a limited number of files and practically creates a whitelist (Listing 6). Even if the process is compromised, the hacker can only do so much, and the scope of the exploits is limited to permissions granted within the profile; in this case, it is limited to certain time-related files.

## Removing Profiles

In most cases, it is not advisable to re-use an app or script, but if the need arises, AppArmor provides a provision to circumvent the scanning process. To remove a profile, you can either delete the profile present in `/etc/apparmor.d/` or put the profile or its soft link under `/etc/apparmor.d/disable`.

After you make the change, notify AppArmor using the following command:

```
apparmor_parser -R profile_path
```

To remove the `ntpd` process from the scanning, use the following command:

```
apparmor_parser -R Z
/etc/apparmor.d/usr.sbin.ntpd
```

If you are deleting a profile, make sure you restart the AppArmor process.

## Roadmap

All active development efforts experience a constant hustle to update and improve. AppArmor strives to add new features – the current version has evolved from its nascent stage, which was

bogged down with huge performance overhead to the kernel/CPU, as it was written in plain text. Version 2 added a compile state machine, which guaranteed execution time by restricting the CPU overload.

Version 3, which is currently in development, will add multiple cache support. Enhanced networking support will bring fine-grained access to networking, which will allow you to monitor network-oriented applications and will also grant access to specific port calls. Support for ZFS database operations, Cgroups, and `chroot` will add additional capabilities for security-minded users.

## Conclusion

MACs, if implemented properly, can provide a very well defined security ring that can render any attack useless or make the damage very predictable. AppArmor creates a whitelist that can ward off any major damage and be independent of user and system, and it creates very lean and simple application-specific rules that are easy to discern and implement.

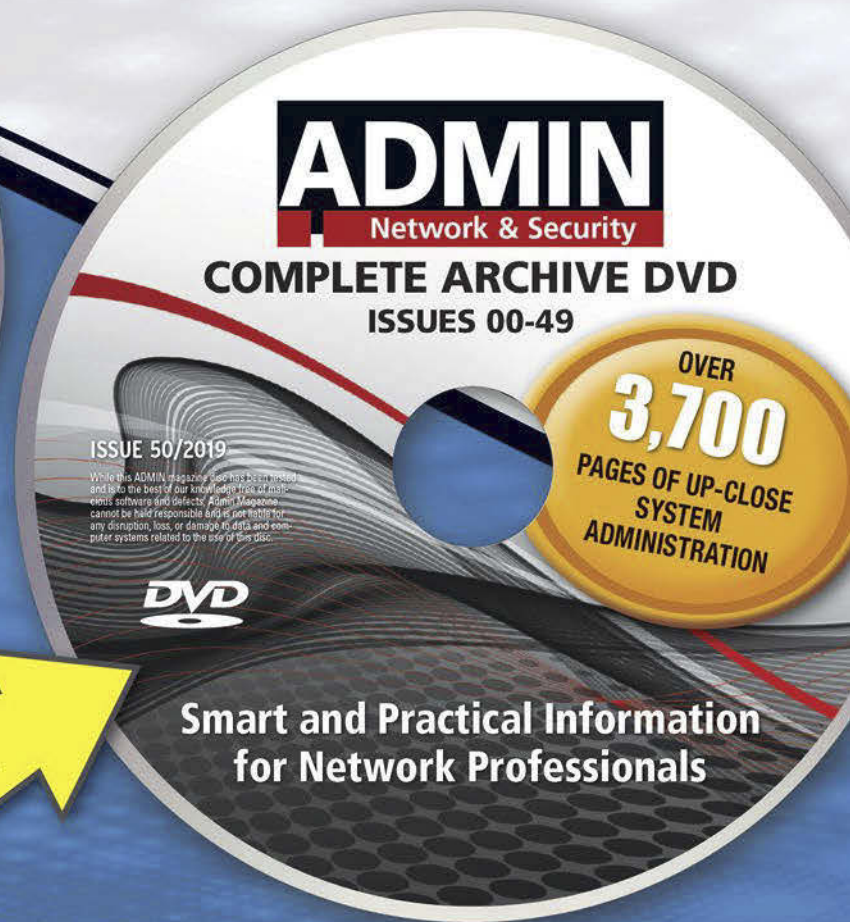
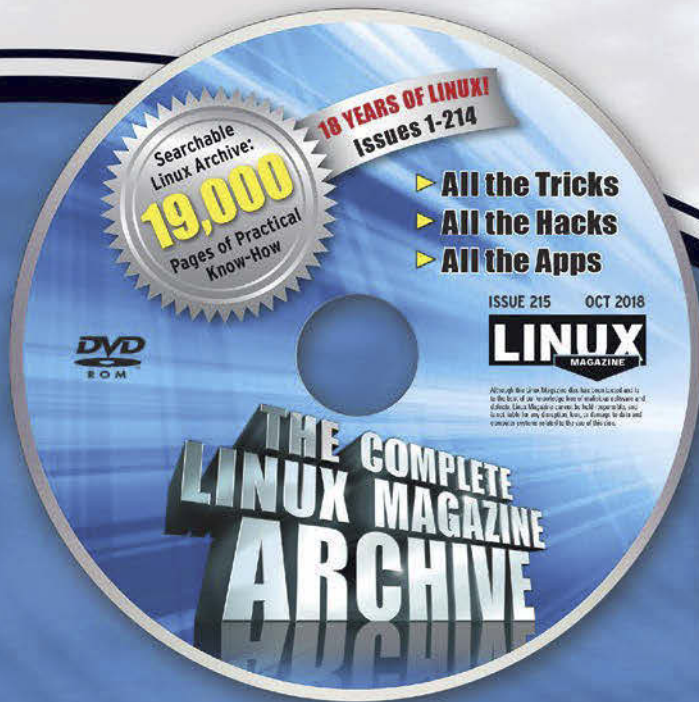
The only gripe with AppArmor is that it offers a limited security solution out of the box. It may work very well in a limited environment, but the need to manually create profiles in a more exhaustive setup can be cumbersome at best. Creating profiles for so many variables can be a daunting task and may ward off interest in using AppArmor in the first place. However, development of new profiles is an ongoing process; more profile features are being added with each passing release.

Does AppArmor guarantee security? Yes and no. Security will always be on an unstable footing, and this article has only scratched the surface towards securing your system. The price of freedom and security is eternal vigilance. ■■■

### Info

- [1] AppArmor Wiki: <https://gitlab.com/apparmor/apparmor/wikis/Documentation>
- [2] Oracle Docs: <https://docs.oracle.com/en/>
- [3] Red Hat Documentation: <https://access.redhat.com/documentation/en-us/>





---

# Complete Your Open Source Library with Archive DVDs!

---

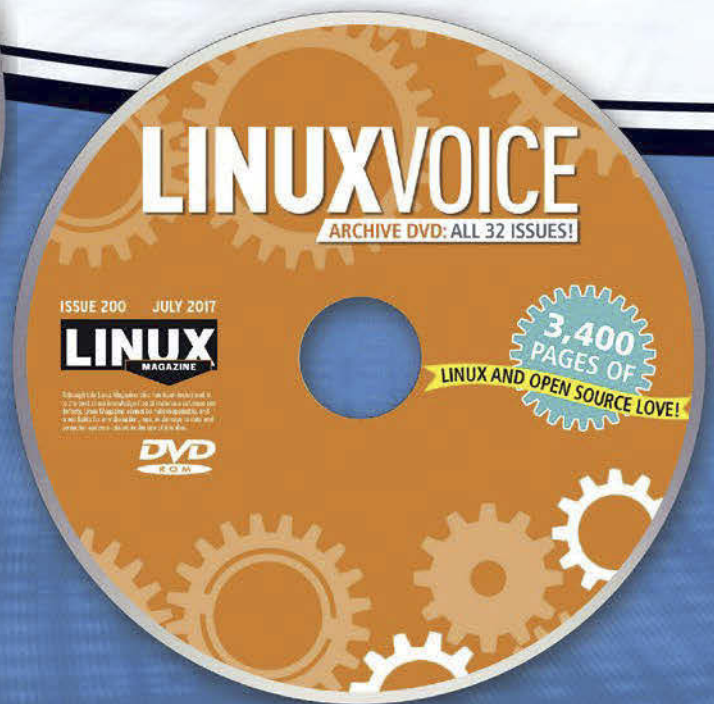
Each fully-searchable archive DVD includes past articles so you can find the content you need quickly.

**Save hundreds off the print and digital copy rate with a convenient archive DVD!**



# Order Your DVD Now!

[shop.linuxnewmedia.com](http://shop.linuxnewmedia.com)





Go program stores directory paths

# Pathfinder

When you change directories at the command line, you often find yourself jumping back and forth between known paths. With a utility written in Go, Mike Schilli records the jumps and shows the way back. *By Mike Schilli*

**W**hile younger coworkers tend to edit their programs with clever IDEs, I still find it most natural to

## Author

**Mike Schilli** works as a software engineer in the San Francisco Bay area, California. Each month in his column, which has been running since 1997, he researches practical applications of various programming languages. If you email him at [mschilli@perlmeister.com](mailto:mschilli@perlmeister.com) he will gladly answer any questions.



jump to local Git repositories with a quick `cd` at the command line and fire up Vim at files with the source code residing there. Typing in the directory path each time is a pain in the ass, and there are usually only half a dozen paths back and forth – so the command line should be able to remember that.

The C shell invented the `pushd` and `popd` commands many years ago, but wouldn't it be much more convenient to automatically record the directories you visit, store them in a database, and even offer search queries for previously visited directories based on criteria such as frequency or the timestamp of the last visit?

In this issue, a Go program by the name of `cdbm` collects the paths accessed by the user during a shell session; the command line user just uses `cd`, and some magic glue in the shell's configura-

tion then taps into the `$PS1` prompt generator. If the directory changes, `cdbm` gets called and stores the new path in an SQLite database on the disk, which later allows search queries whose results can be accessed directly by the user for navigation help. Bash users can modify their `.bashrc` file to enable this. On typing a newly introduced command `c`, users will see a selection list with the last directories visited (Figure 1). After selecting one of them with the cursor keys and pressing Enter, the shell directly jumps there (Figure 2).

If the list of hits grows beyond the preconfigured set limit of five entries,

```
mybox.mschilli:~$ c
Use the arrow keys to navigate: ↓ ↑ → ←
? Pick a directory:
  /home/mschilli/git/articles/cdbm
  ▶ /home/mschilli/git/articles/cdbm/eg
  /home/mschilli/git/articles/go-testing
  /home/mschilli/git/articles/dateday
  ↓ /home/mschilli/git/articles/gotui
```

**Figure 1:** The `c` command shows you the last directories you visited ...

```
✓ /home/mschilli/git/articles/cdbm/eg
mybox.mschilli:eg$ █
```

**Figure 2:** ... and jumps to the one you select.



the nifty terminal UI (as shown in Figure 1) displays a small down arrow, indicating that the user can move the cursor further down to reveal previously hidden entries. So how does this work?

## Hitchhiking a Prompt Ride

Once the Bash shell has executed a command, it generates the line prompt so that the user knows that it is his turn again. Instead of boring \$ or # characters, experienced shell users often define individual prompts in the \$PS1 variable; they can display the username, the hostname, and the current directory. For example, the following statement

```
export PS1='\h.\u:\W$ '
```

defines a prompt with the hostname (\h), a separating dot, the username (\u), a separating colon, the current directory, a dollar sign, and a space. On my machine in the git directory, this comes up as:

```
mybox.mschilli:git$
```

Now the \$PS1 prompt variable does not just support the placeholders used above, which it replaces with current values, but also commands to be executed, whose output it interpolates into the prompt string:

```
export PS1='${(cdbm -add)\h.\u:\W}\$ '
```

This definition tells Bash to call the cdbm program with the -add option after every shell command executed. cdbm is the Go program in Listing 1 [1] that determines the current directory in -add mode and stores the path with the current timestamp in a table of an automatically created SQLite single file database. If the path already exists, cdbm only refreshes the timestamp for the entry. While the user keeps changing directories with cd, paths with timestamps accumulate in the database (Figure 3).

Of course cdbm -add does not output anything, but returns without comment

### Listing 1: cdbm.go

```
001 package main
002
003 import (
004     "database/sql"
005     "flag"
006     "fmt"
007     "github.com/manifoldco/promptui"
008     _ "github.com/mattn/go-sqlite3"
009     "os"
010     "os/user"
011     "path"
012 )
013
014 func main() {
015     addMode := flag.Bool("add", false,
016         "addition mode")
017     flag.Parse()
018
019     db, err :=
020         sql.Open("sqlite3", dbPath())
021     panicOnError(err)
022     defer db.Close()
023
024     _, err = os.Stat(dbPath())
025     if os.IsNotExist(err) {
026         create(db)
027     }
028
029     dir, err := os.Getwd()
030     panicOnError(err)
031
032     if *addMode {
033         dirInsert(db, dir)
034     } else {
035         items := dirList(db)
036         prompt := promptui.Select{
037             Label: "Pick a directory",
038             Items: items,
039         }
040
041         _, result, err := prompt.Run()
042         panicOnError(err)
043
044         fmt.Fprintf(os.Stderr,
045             "%s\n", result)
046     }
047 }
048
049 func dirList(db *sql.DB) []string {
050     items := []string{}
051
052     rows, err := db.Query(`SELECT dir FROM
053     dirs ORDER BY date DESC LIMIT 10`)
054     panicOnError(err)
055
056     usr, err := user.Current()
057     panicOnError(err)
058
059     for rows.Next() {
060         var dir string
061         err = rows.Scan(&dir)
062         panicOnError(err)
063         items = append(items, dir)
064     }
065
066     if len(items) == 0 {
067         items = append(items, usr.HomeDir)
068     } else if len(items) > 1 {
069         items = items[1:] // skip first
070     }
071
072     return items

```



## Listing 1: cdbm.go (continued)

```

073 }
074
075 func create(db *sql.DB) {
076     _, err := db.Exec(`CREATE TABLE dirs
077         (dir text, date text)`)
078     panicOnErr(err)
079
080     _, err = db.Exec(`CREATE UNIQUE INDEX
081         idx ON dirs (dir)`)
082     panicOnErr(err)
083 }
084
085 func dirInsert(db *sql.DB, dir string) {
086     stmt, err := db.Prepare(`REPLACE INTO
087         dirs(dir, date)
088         VALUES(?, datetime('now'))`)
089     panicOnErr(err)
090
091     _, err = stmt.Exec(dir)
092     panicOnErr(err)
093 }
094
095 func dbPath() string {
096     var dbFile = ".cdbm.db"
097
098     usr, err := user.Current()
099     panicOnErr(err)
100     return path.Join(usr.HomeDir, dbFile)
101 }
102
103 func panicOnErr(err error) {
104     if err != nil {
105         panic(err)
106     }
107 }

```

after the work is done, so that the `$PS1` prompt defined above remains the same, even if the Bash shell secretly called the directory butler while composing the prompt.

## Here We Go

To compile Listing 1, the following command sequence generates a new Go module in the same directory where the build process happens later on:

```

go mod init cdbm
go build

```

Listing 1 references a number of useful Go packages on GitHub, which the call to `go build` automatically retrieves as source code, because of the previous module definition, and compiles as libraries before compiling Listing 1. The resulting `cdbm` binary contains everything, including a driver for creating and querying SQLite databases.

Once you have copied the binary to a location where the shell can find it in the search `$PATH`, you have to change two things in the `.bashrc` bash profile in order to benefit from the new utility. First, add the `$PS1` definition from above and second, define a Bash function `c` that calls `cdbm` in selection mode and later outputs the path the user selected:

```

export PS1='${(cdbm -add)\h.\u:~\W}$ '
function c() { dir=$(cdbm 3>&1 2
1>&2 2>&3); cd $dir; }

```

Now, if you type `c` after `.bashrc` has run (either automatically when opening a new shell or manually via `source .bashrc`) in the shell, the newly defined bash function `c` above will call the `cdbm` program. The latter writes the selection list to `stdout`, the user then interacts with the cursor keys, selects a directory with `Enter`, and `cdbm` writes the result to `stderr`.

Now the function only has to pass the contents of `stderr` (the selected path) to the shell's `cd` function, which then changes to the specified directory. This is easier said than done, because `cd` is not a program, but a built-in shell function. A program could change its own working directory, but not that of the parent process, the shell itself. To complicate matters, unlike other Unix

```

mybox.mschilli:eg$ sqlite3 ~/.cdbm.db
SQLite version 3.19.3 2017-06-27 16:48:08
Enter ".help" for usage hints.
sqlite> .schema
CREATE TABLE dirs (dir text, date text);
CREATE UNIQUE INDEX idx ON dirs (dir);
sqlite> select * from dirs;
/tmp|2019-07-13 16:34:51
/home/mschilli/git/articles/go-geosearch|2019-07-13 16:34:52
/home/mschilli/git/articles/mitmproxy|2019-07-13 16:34:53
/home/mschilli/git/articles/go-scraper|2019-07-13 16:34:54
/home/mschilli/git/articles/gimp-python|2019-07-13 16:34:55
/home/mschilli/git/articles/pfsense-scraper|2019-07-13 16:34:56
/home/mschilli/git/articles/video|2019-07-13 16:34:57
/home/mschilli/git/articles/inline|2019-07-13 16:34:58
/home/mschilli/git/articles/contest-2019-debrief|2019-07-13 16:34:59
/home/mschilli/git/articles/gimp|2019-07-13 16:35:00
/home/mschilli/git/articles/contest-2019|2019-07-13 16:35:01
/home/mschilli/git/articles/findex|2019-07-13 16:35:02
/home/mschilli/git/articles/progressbar|2019-07-13 16:35:03
/home/mschilli/git/articles/go-concurrency|2019-07-13 16:35:04
/home/mschilli/git/articles/gotui|2019-07-13 16:35:05
/home/mschilli/git/articles/dateday|2019-07-13 16:35:07
/home/mschilli/git/articles/go-testing|2019-07-13 16:35:08
/home/mschilli/git/articles/cdbm|2019-07-13 16:35:17
/home/mschilli|2019-07-13 16:37:33
/home/mschilli/git/articles/cdbm/fig|2019-07-13 16:37:35
/home/mschilli/git/articles/cdbm/eg|2019-07-13 16:37:43
sqlite> █

```

Figure 3: The SQLite database stores paths recently traveled along with timestamps.

commands, `cd` insists on being provided with an actual argument, the directory, which it cannot read from `stdin` by way of a pipe.

This explains the wild trick the Bash function resorts to above. After calling `cdm` it swaps its `stdout` and `stderr` channels. To do this, it first uses `3>&1` to define a file descriptor named `3` and points it to the same channel as the file descriptor `1` (i.e., `stdout`). The following redirection `1>&2` assigns a new value to the `1` descriptor and points it to a descriptor `2` (i.e., `stderr`). The third – that is `2>&3` – assigns the value of the temporarily used file descriptor `3` (i.e., the cached `stdout`) to `stderr`. In other words, the terminal UI’s output of `cdm` no longer ends up in `stdout`, but instead in `stderr`, and the result of the selected directory is sent to `stdout`. We in engineering call this a “switcheroo.”

The `dir=$(...)` construct then grabs `stdout` and assigns it to the `$dir` variable. The following `cd` statement for the directory change, separated by a semicolon, receives the value from the variable and jumps to the specified directory. This whole rigmarole was necessary, because the easy way of capturing `stdout` does not work, as the terminal UI insists on writing to it, and redirecting it would leave the user without any visual output with which to interact.

## Nitty Gritty

The `cdm.go` program in Listing 1 only has to do two things. First, if the `-add` option is present, it stores the current working directory in the SQLite database. If the option is not set, it displays the terminal UI with the SQLite entries, lets the user select one, and outputs the chosen path to `stderr`.

To do so, it defines the `-add` option with the help of the standard `flag` package in line 15. If `cdm` is called with `-add`, the pointer value dereferenced with `*addMode` has a true value after parsing the command-line arguments with `flag.Parse()`, and line 33 branches to the function `dirInsert()` starting in line 85. In display mode, the `else` branch starting in line 34 uses `dirList()` to fetch all the paths stored in the SQLite database and sorts them in descending order of the dates on which they were added.

The terminal UI for selecting a directory gets drawn by the `promptui` package, which offers `Select()` and `Run()` functions to configure the list and then switch to user interaction mode. The result, the path selected by the user as a string, is finally output to `stderr` by lines 44 and 45, whereupon the program terminates.

## Database Voodoo

A new directory is added to the database with `dirInsert()` as of line 85. If the SQLite database does not yet exist (Listing 1 checks for the existence of the database file), line 76 builds a new one and creates a fresh `dirs` table in it with the SQL `create` command. The table’s two columns, `dir` and `date`, are of the text type. The fact that the directory path is a text string is not surprising, but SQLite also saves dates as text and compares them in string mode, which works because the timestamps are in the format `YYYY-MM-DD HH::MM::SS`, so later times also alphanumerically follow earlier ones.

We do not want SQLite to generate a new table row for existing paths, but simply refresh the existing entry’s timestamp. This could be solved in SQL by a separate `select` query, followed by case logic, but in the SQLite dialect this is done more elegantly with the special `REPLACE` function (line 86).

This works in a similar way to SQL’s standard `UPDATE` function, except it creates missing entries – but only if a unique index is defined for the corresponding table column. This explains why line 80 adds an index to the `dir` column after the table definition, causing `replace` in line 86 to create new entries and refresh old ones as needed.

The `dirList()` function retrieves existing database entries. The `select` statement in line 52 sorts them in descending order according to insertion date (i.e., entries created shortly beforehand appear at the top of the selection list). The `LIMIT 10` statement fetches a maximum of 10, but since the terminal list display scrolls down on request, you could just as easily leave this out.

The for loop starting in line 59 uses `rows.Next()` and `rows.Scan()` to fetch the search query’s next hits; the `append` statement in line 63 appends them to the `items` array slice. If the database is still

unpopulated, line 67 inserts the user’s home directory; otherwise, the displayed selection list would be empty and leave the user confused.

If two or more hits are found, line 69 steals the first one and removes it from the list, because this is the entry of the last directory visited (i.e., the current directory to which the user will probably not want to jump). The `dbPath()` function starting in line 95 specifies the path to the SQLite file in which the data is stored, this is hard-coded in Listing 1 as `~/cdbm.db` in the user’s home directory.

## Quite Wordy

It’s worth noting that a Go program, which actually doesn’t contain so much logic, needs a large number of lines. Go’s relentlessly required, explicit error handling of each return value is partly to blame. An exception handler would be more compact for such a simple utility. The `panicOnErr()` function starting in line 103, which checks an error value passed to it and immediately aborts the program by calling `panic()`, helps to save lines. It is rumored that the next Go version will accommodate program authors with more compact mechanisms.

## More Convenience

For hobbyists, the fun only starts here. For instance, the script could be extended to include a search function that only offers paths that match a search term entered at the command line. For example, the user could enter `c usr` and only see paths containing the string `usr`. And since all the user data are stored in an SQLite database whose schema can be easily extended, it makes sense to assign a counter to each stored path, which `cdm` increments by one each time a directory is visited.

This could shift frequently visited paths higher up in the selection list based on an algorithm, because users would not want to scroll a long way to access frequently used paths. And, who knows, maybe it’s worth adding a dash of artificial intelligence? A self-learning directory butler would be attractive to my younger coworkers. ■■■

## Info

- [1] Listings for this article:  
<ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/228>



Cleaning up the filesystem

# Unification

At long last, Debian joins other Unix-like distributions in merging /usr directories. *By Ferdinand Thommes*

Since 1993, the Filesystem Hierarchy Standard (FHS) [1] has been the guideline for Unix-like directory structures. It requires the root directory partition to contain all the files the system needs for booting and mounting additional partitions.

This results in 14 directories or symbolic links to them: /bin, /boot, /dev, /etc, /lib, /media, /mnt, /opt, /run, /sbin, /srv, /tmp, /usr, and /var. Only the /opt,

/usr, and /var folders can be located on other partitions. In addition, FHS considers the home directory to be optional, along with /root, /lib32, and /lib64.

In 2015, FHS was integrated into the Linux Standard Base (LSB). FHS is now maintained under the umbrella of the Linux Foundation.

Starting in 2012, several distributions have chosen to merge the /usr directories [2], thus modifying the FHS (and are therefore no longer LSB-compliant). Starting with Debian 10 Buster, Debian has now joined the /usr merge ranks.

## Root Canal Treatment

In Debian 10 Buster, Debian makes changes to the root directory for new installations. The /usr merge aims to move the contents of the /bin, /sbin, /lib, and /lib64 directories to matching directories below /usr. For compatibility reasons, only symbolic links remain in the root directory. Thus, two directories for binary files and two directories for libraries are merged into one (Figure 1).

Debian comes to the /usr merge party relatively late compared to other distributions. As early as 2012, Fedora system

developers Harald Hoyer and Kay Sievers implemented the /usr merge in Fedora 17 [3]; Solaris took the leap two years earlier with version 11. Arch Linux also started unifying the directories in 2012 and merging the /usr directories in 2013. Arch Linux also merged /bin and /sbin, which Debian doesn't intend to do at this time.

Lennart Poettering, creator of PulseAudio, Avahi, and systemd, has supported this change, because it facilitates stateless systems [4]. Following the /usr merge, it is possible to mount the immutable parts of the installed operating system with write protection, to update them atomically, or even to distribute them to several hosts or containers.

Before the /usr merge, historical partitioning was needed to mount /usr as a separate partition or across a network. The system mounts the /usr directory early on in the boot process, long before the initial RAM disk (initrd) [5].

## Historically Conditioned

The roots of the historical division, where some directories are duplicated both di-

```

/
|-- etc
|-- usr
|   |-- bin
|   |-- sbin
|   |-- lib
|   `-- lib64
|-- run
|-- var
|-- bin -> usr/bin
|-- sbin -> usr/sbin
|-- lib -> usr/lib
`-- lib64 -> usr/lib64

```

**Figure 1:** This is how the /usr merge is represented schematically in the file tree.



rectly below root and in /usr, date back to the beginning of Unix development in the 1970s. When Ken Thompson and Dennis Ritchie started developing Unix in 1969, storage space was limited.

In the beginning Thompson and Ritchie used floppy disks and then switched to a Digital Equipment Corporation [6] PDP-11 computer in about 1971. The PDP-11 already had a hard disk with a capacity of up to 512KB and supported a removable RK05 DECpack [7] magnetic media drive with 2.5MB, which was larger but far slower.

When the Unix root system grew beyond half a megabyte, the largest directory with the applications and tools was stored on the RK05-DECpack, which until then had contained only user data. Based on this practice, they named the mount point /usr (other sources claim that the acronym stands for “Unix System Resources”). Probably both are correct, because after the introduction of a third hard disk, the user data migrated there, making renaming meaningful.

Since the root directory was now distributed over two hard disks due to space constraints, a successful boot process required that everything on the first disk be kept accessible in order to boot the system to an extent where /usr could be mounted. This led to directories such as /bin, /lib, and /sbin being found both directly below the root and again below /usr.

Despite the advent of hard disks with sufficient capacity for the entire root system, this setup continued for many years and flowed into the FHS. Today, there is no reason to duplicate these directories. (More information on this topic can be found in an essay by Rob Landley [8]).

## Penultimate

Canonical has delivered a unified /usr directory since Ubuntu 19.04. With the Debian Buster release, openSUSE will be the only major distributions that is not merging the /usr directories.

Debian started the migration in 2016, when the *usrmerge* package became available in the unstable branch [9]. *usrmerge* contains a Perl script that converts

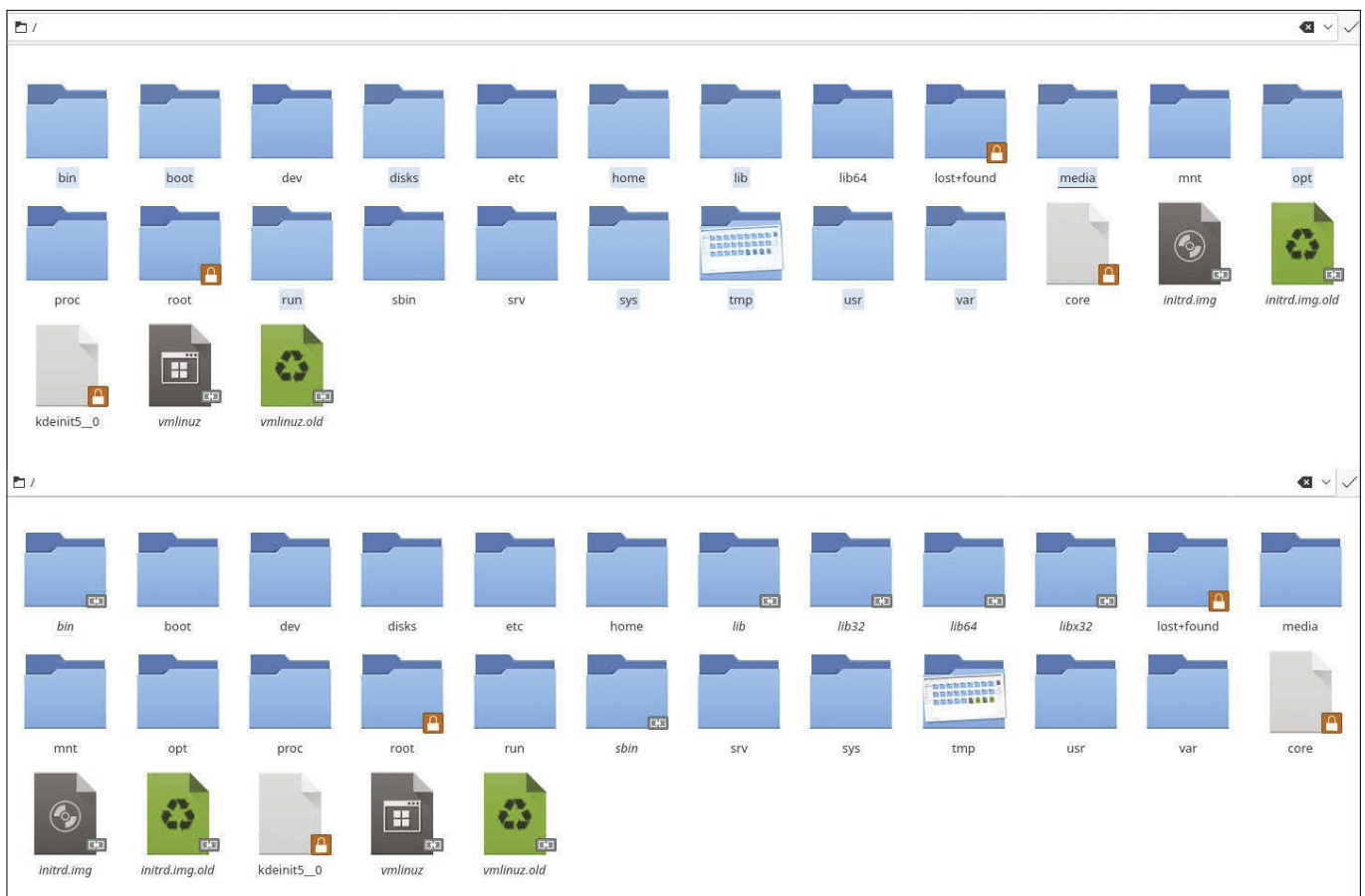
an existing system into one with a merged /usr directory. The developers also made a change to the *debootstrap* package so that it creates the required symbolic links itself before installing packages in a chroot.

Due to bugs that could not be fixed in time for Debian 9, the developers shifted the unification of / and /usr to Debian 10. However, during further development, other errors were made since packages created after unification would not run on unconverted systems.

Most of these bugs are now a thing of the past, or patches are available [10]. However, Debian developer Ian Jackson believes that the change could lead to more serious bugs and called on the Debian Technical Committee (TC) to postpone the /usr merge again [11]. In March, however, the TC rejected this request, so Debian 10 features the completed /usr merge [12].

## “Testing” Reality

As an explanation, the TC discussed the advantages and disadvantages, as well as



**Figure 2: Before /usr merge (above): The /bin, /lib, /lib64, and /sbin directories reside directly below root and are duplicated in the /usr directory. After the conversion (below): The /bin, /lib, /lib32, /lib64, and /sbin directories exist as symbolic links to the physical directories in /usr.**

the future planning for Debian 11 Bullseye. People who install Debian 10 Buster or derivatives thereof can look forward to a system with a unified /usr directory and symbolic links of /bin, /sbin, /lib to the /usr/bin, /usr/sbin, and /usr/lib directories (Figure 2).

The setup does not affect existing systems unless this is desired by the user. In this case, you just need to install the *usrmerge* package, which moves the corresponding directories and makes further adjustments. During the install, Debian explicitly asks for confirmation, since the change cannot be reversed.

After the change, check to see whether the `/etc/dpkg/dpkg.cfg.d/usrmerge` directory exists and if it contains package names. If it does, this means that there are still conflicts with these remaining packages. If it is empty or does not exist, you can remove *usrmerge*; otherwise wait until the list is empty (Figure 3).

Unlike Debian, other distributions switched to a unified /usr directory on deadline. Debian's transition issues largely resulted from the decision by developers to leave older systems untouched and give users a choice. For an

in-depth analysis of why Debian has had more problems with this migration than other distributions, see LWN.net [13].

## Conclusions and Outlook

Switching to a merged /usr directory eliminates the historical duplication of directories and paves the way for modern techniques, such as stateless systems and atomic updates, thus facilitating the

handling of snapshots. However, most desktop users are unlikely to notice any changes. The symbolic links to the sub-directories in /usr, on the other hand, will probably remain with us for some years. For Debian 11, the aim is for Debian packages that can be built on systems with and without the /usr merge and that still will work on the other system without restrictions. ■■■

### Info

- [1] FHS: [https://en.wikipedia.org/wiki/Filesystem\\_Hierarchy\\_Standard](https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard)
- [2] The /usr merge: <https://wiki.debian.org/UsrMerge>
- [3] Fedora 17: <https://www.freedesktop.org/wiki/Software/systemd/TheCaseForTheUsrMerge/>
- [4] Stateless systems: [https://en.wikipedia.org/wiki/Stateless\\_protocol](https://en.wikipedia.org/wiki/Stateless_protocol)
- [5] initrd: [https://en.wikipedia.org/wiki/Initial\\_ramdisk](https://en.wikipedia.org/wiki/Initial_ramdisk)
- [6] PDP-11: <https://en.wikipedia.org/wiki/PDP-11>
- [7] RK05: <https://en.wikipedia.org/wiki/RK05>
- [8] Rob Landley: <http://landley.net/writing/hackermothly-issue022-pg33.pdf>
- [9] usrmerge package: <https://salsa.debian.org/md/usrmerge/raw/master/debian/README.Debian>
- [10] Bugs: <https://bugs.debian.org/cgi-bin/pkgreport.cgi?tag=usrmerge;users=md@linux.it>
- [11] Ian Jackson's bug report: <https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=%23914897>
- [12] TC decision: <https://lists.debian.org/debian-devel-announce/2019/03/msg00001.html>
- [13] Debian migration issues: <https://lwn.net/Articles/773342/>

```

root@dev:~# ls -la /
insgesamt 76
drwxr-xr-x 19 root root 4096 Mai 19 08:36 .
drwxr-xr-x 19 root root 4096 Mai 19 08:36 ..
lrwxrwxrwx 1 root root 7 Jan 11 21:48 bin -> usr/bin
drwxr-xr-x 4 root root 4096 Mai 16 11:33 boot
drwxr-xr-x 21 root root 3740 Mai 16 18:10 dev
drwxr-xr-x 2 root root 4096 Jan 25 19:38 disks
drwxr-xr-x 157 root root 12288 Mai 19 08:29 etc
drwxr-xr-x 3 root root 4096 Jan 25 19:40 home
lrwxrwxrwx 1 root root 44 Mai 16 11:14 initrd.img -> boot/initrd.img-5.1.2-towo.2-siduction-amd64
lrwxrwxrwx 1 root root 44 Mai 16 11:14 initrd.img.old -> boot/initrd.img-5.0.5-towo.1-siduction-amd64
lrwxrwxrwx 1 root root 7 Jan 11 21:48 lib -> usr/lib
lrwxrwxrwx 1 root root 9 Jan 11 21:48 lib32 -> usr/lib32
lrwxrwxrwx 1 root root 9 Jan 11 21:48 lib64 -> usr/lib64
lrwxrwxrwx 1 root root 10 Jan 11 21:48 libx32 -> usr/libx32
drwx----- 2 root root 16384 Jan 25 19:31 lost+found
drwxr-xr-x 4 root root 4096 Mär 14 10:06 media
drwxr-xr-x 2 root root 4096 Jan 25 19:38 mnt
drwxr-xr-x 8 root root 4096 Feb 7 18:03 opt
dr-xr-xr-x 275 root root 0 Mai 16 18:10 proc
drwx----- 12 root root 4096 Mai 19 08:34 root
drwxr-xr-x 37 root root 1060 Mai 19 08:42 run
lrwxrwxrwx 1 root root 8 Jan 11 21:48/sbin -> usr/sbin
drwxr-xr-x 2 root root 4096 Jan 11 21:48 srv
dr-xr-xr-x 13 root root 0 Mai 16 18:10 sys
drwxrwxrwt 24 root root 840 Mai 19 09:20 tmp
drwxr-xr-x 14 root root 4096 Jan 11 21:52 usr
drwxr-xr-x 11 root root 4096 Jan 11 21:48 var
lrwxrwxrwx 1 root root 41 Mai 16 11:14 vmlinuz -> boot/vmlinuz-5.1.2-towo.2-siduction-amd64
lrwxrwxrwx 1 root root 41 Mai 16 11:14 vmlinuz.old -> boot/vmlinuz-5.0.5-towo.1-siduction-amd64
-rw-r--r-- 1 root root 0 Jan 25 19:29 .xorgconfig-was-here
root@dev:~#

```

Figure 3: The root directory in the terminal also reflects the new relationships and displays the symbolic links.



# EXPERT TOUCH

After selling out in 2018, the new *Linux Shell Handbook* is available now! The 2019 edition is packed with utilities for configuring and troubleshooting systems.

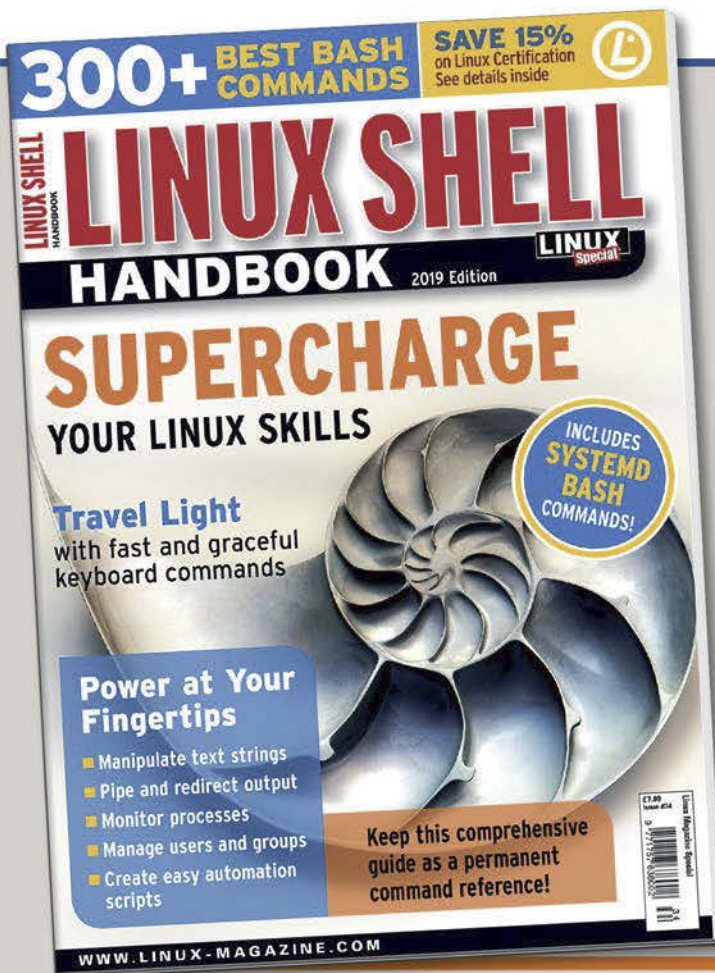
**2019  
Edition!**

The *Shell Handbook* provides a comprehensive look at the world inside the terminal window. You'll learn to navigate, manipulate text, work with regular expressions, and customize your Bash settings.

**Exclusive Bonus:** Each issue includes a special discount saving you **15% off LPIC-1 certification** from Linux Professional Institute (LPI). Look for the special code inside your copy!

Here's a look at just some of what you'll see in the new *Shell Handbook*:

- Customizing Bash
- Pipes and Redirection
- Regular Expressions
- Text Manipulation Tools
- Systemd
- Bash Scripting
- Networking Tools
- And much more!



Make the *Shell Handbook* your permanent desktop reference on the world of the terminal window.

ORDER ONLINE:  
[shop.linuxnewmedia.com/specials](http://shop.linuxnewmedia.com/specials)





# MakerSpace

Making smart devices  
smarter with Tasmota

## Home Rule

Flashing IoT devices with new firmware lets you wield control and keep them out of the cloud. *By Chris Dock*

**I**n the age of robotic vacuums, robotic lawn mowers, and mopping robots, I want to add automation and intelligence to my apartment. Just the act of purchasing a smart socket that has an Android app makes this both an easy and a comfortable solution for people wanting to add a touch of home automation to their living space.

However, I am paranoid enough that when devices are in my network, I want to have full control over them. It bothers me that this new functionality comes with the “additional feature” of these devices calling home to a server somewhere on the Internet. I was optimistic that I could achieve my goal of automation but also keep full control over my own devices.

To achieve this control, I was hoping I could purchase a few smart adapters and find one that I could bend to my will. The good news is that the security on each of these devices was good enough that I wasn’t able to circumvent it with a playback or man-in-the-middle attack, but I didn’t actually have any luck gaining complete control of the devices, so I set out to find a way to modify an existing device to achieve my goals.

Little did I know that this is a fairly active area of development in the open source community. The solution is to flash the devices with new firmware that provides enhanced functionality. I

looked into a couple of different solutions under active development. The oldest and presumably most mature firmware solution is Tasmota.

### Tasmota

The Tasmota project [1] contains additional functionality to support the MQTT (Message Queuing Telemetry Transport) protocol, so your device can be an MQTT client and publish and receive messages.

The list of main functions supported by Tasmota is pretty impressive:

- Support for the MQTT protocol
- Support for HTTP requests
- Support for most common sensors
- Over-the-air updates
- Belkin Wemo and Philips Hue emulation
- Timers
- Domoticz integration
- ESPTool FTDI programmer/serial programmer

My needs are pretty modest; I simply want to connect the device to my home network. However, in addition to all the other functionality, I get another cool feature for free. Tasmota offers over-the-air (OTA) updates, which means once you have flashed the device, you won’t have to unplug, disassemble, flash, and reassemble each time you want to upgrade the firmware, which is priceless for anyone unfamiliar with the technology.

Before using this firmware, you need to have all the necessary tools and materials:

- FTDI serial converter
- Header pins
- Soldering iron and solder
- Firmware
- Low-melting-point solder (optional)

The device I chose to modify was the Sonoff S20 (Figure 1), an extremely friendly device from the perspective of the home do-it-yourselfer. The circuit board is big, is easily accessible, and has been designed so it can be enhanced with connector pins to allow external programming.

Taking apart the Sonoff was refreshingly easy. None of the screws are hidden behind labels, and the case comes apart and can be put together much like playing with legos. No blobs of glue prevent access to any part of it. Inside the case, the printed circuit board (PCB) is also held in place with screws, and it appears the overall design was done to simplify the assembly for multiple markets. The design of the S20 separates the power socket from the circuit board in such a way that plugging in your power cord doesn't put any stress on the PCB or any of its connections. The manufacturer, iTead, must feel pretty confident in their product, because they also made the schematic available on their website [2].



**Figure 1:** Sonoff S20.

Not only is it easy to disassemble and reassemble the case, the PCB that controls the power socket is ready to be programmed. The only step necessary is to solder on some header pins so you can flash the Sonoff with the new software (Figure 2). Once the headers are soldered to the PCB, you can then use your FTDI adapter male or female header pins, depending on your FTDI adapter and cable setup.

Soldering the header to the board should take just a matter of minutes, but before you proceed, you might want to take some notes. Some people have been reporting that their circuit board already has markings describing what each pin is. That was not the case for my board, although the VCC (power in) pin did have a small arrow pointing to it. This pin is also visibly different from the rest: The through-hole plating on the board has a square outline around the hole. This differentiation, as well as the pinout for the header pins, is clearly marked in Figure 2.

During this project you should keep two important points in mind. The first and most important is to make sure your device is not plugged into the wall, so it is easier to work on the device and the chance of electrocuting yourself or blowing up your laptop is considerably reduced. The second point is that the power from your FTDI connector needs to be 3.3V. Some FTDI adapters use 5V, so if you are not certain how much

power your controller uses, look it up online or purchase an inexpensive 3.3V connector for this project.

## Getting Your Hands Dirty

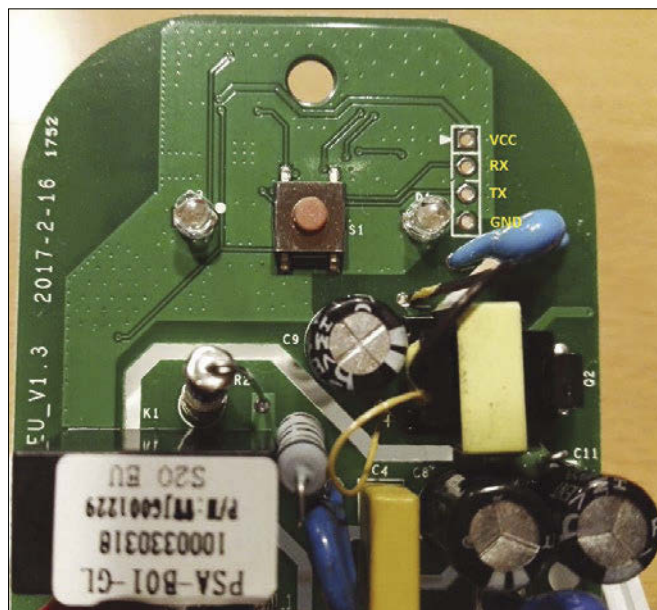
The Sonoff, like most smart devices, will be powered by a microcontroller, which is almost as good as a CPU. Albeit a bit slower, it sips energy and, as a friend used to say, they are cheap as potato chips.

The only downside of devices powered by microcontrollers is that the resources are usually somewhat limited, as well. The limiting factor that restricts the size of the firmware and ultimately the functionality is the number of resources available. The Sonoff that I purchased has only 1MB of flash memory, which as small as it sounds, is plenty for controlling the smart socket (see the “Querying Memory Size” box). However, if the firmware occupies more than half of the available memory, OTA updates are no longer possible.

The convenience of OTA firmware flashing should not be underestimated. Disassembling a smart socket, flashing the firmware, and reassembling it is not a big deal unless you have 20 of them. The only thing preventing OTA updating is the size of the firmware relative to the size of free memory. A number of other trailblazers on the Internet have already found a solution to this problem, which is to get a larger flash chip to hold the program.

This small upgrade only requires a few minutes to desolder the old chip and solder in a new chip.

Although this solution sounds radical, it isn't that much more so than opening up a device and soldering header pins onto it. The PN25F08B that comes with the Sonoff S20 is just a 1MB flash chip that can be replaced with a larger memory chip. The most



**Figure 2:** Header pins with VCC, Rx, Tx, and GND labels added.

Querying Memory Size

Memory size might become more important in the future, with different devices being manufactured with different memory sizes. Thus, it is important to know how much memory exists in the device for either backing up or determining the maximum size for a firmware image.

This information can be queried from the device itself:

```
$ sudo ./esptool.py --port /dev/ttyUSB0 flash_id
esptool.py v2.6
Serial port /dev/ttyUSB0
Connecting...
Detecting chip type... ESP8266
Chip is ESP8266EX
Features: WiFi
MAC: 84:f3:eb:b0:6e:07
Uploading stub...
Running stub...
Stub running...
Manufacturer: 5e
Device: 4014
Detected flash size: 1MB
Hard resetting via RTS pin?
```

The value (shown in the next-to-the-last line) is normally 1MB for the S20, but it might be different for other hardware (Figures 3 and 4).

```
Stub running...
Manufacturer: 5e
Device: 4014
Detected flash size: 1MB
Hard resetting via RTS pin...
```

Figure 3: Querying the memory before replacing the memory chip.

```
Stub running...
Manufacturer: ef
Device: 4016
Detected flash size: 4MB
Hard resetting via RTS pin...
```

Figure 4: Querying the memory after replacing the memory chip.

common replacement seems to be the 4MB Winbond 25Q32FVSI.

To make the replacement, carefully heat the legs of the chip and remove

(Figure 5). This process is easier if you have a hot air gun or, alternatively, some low-melting-point solder wire.

The reports I have read said the new memory chip was the same size, so no unusual hacks are necessary (e.g., straightening or bending the pins to odd angles). In my case, the replacement chip was slightly larger

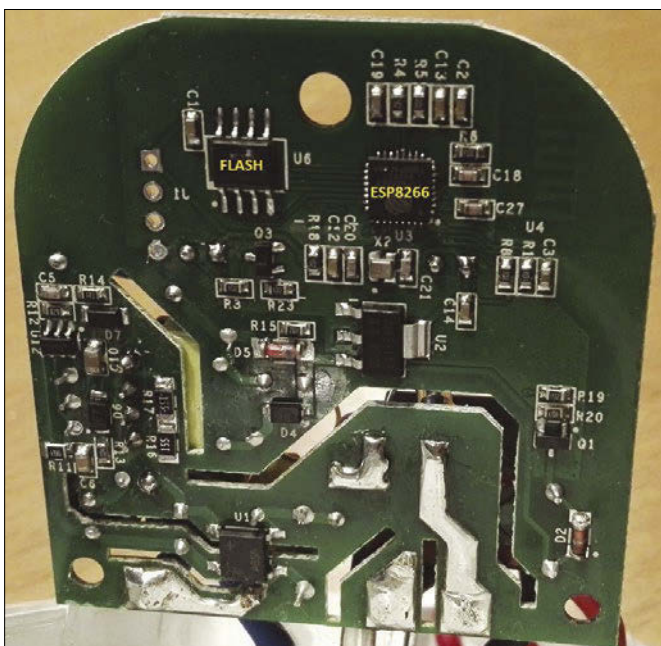


Figure 5: Flash memory chip.

(Figure 6), but it still (barely) fit onto the pads.

Installing the Firmware

A compiled copy of just the Tasmota firmware can be downloaded from GitHub [3], or you can download the source code if you want to take a closer look or compile it yourself. I already had a copy of esptool.py [4] on my laptop. Flashing the firmware is described in full detail in the “Flashing a Device” box. Once you have everything ready to flash the firmware, including the firmware, you are ready to begin.

If your device has 1MB of memory, then backing up is just a matter of running the command:

```
sudo ./esptool.py --port /dev/ttyUSB0 read_flash 0 0x100000 flash_contents.bin
```

It doesn’t matter whether ESPTool is being used for reading memory, gathering information, or writing new firmware: The completion of each action will cause the device to leave programming mode.

The original firmware on the Sonoff allows it to be controlled by the eWeLink application from either the Apple App Store or Google Play. Because flashing new firmware will overwrite the existing firmware, you should back up before continuing. If you later change your mind, you can always restore the device to its original state so it can be controlled by the eWeLink app.

Remember that the original firmware is tied to the device and cannot be flashed into another Sonoff device.

Configuring WiFi

Which device or which firmware is running is immaterial if it is not reachable over your network. Most Internet of Thing (IoT) devices treat this problem

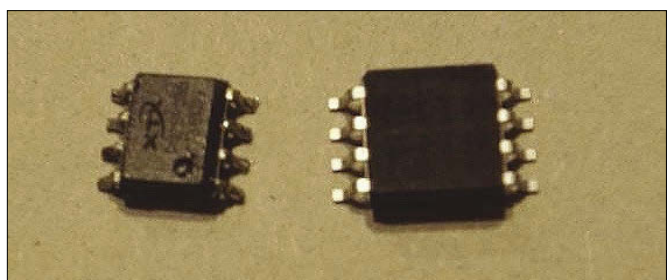


Figure 6: The old memory chip (left) compared with the new, larger chip (right).



## Flashing a Device

To flash the firmware, the FTDI adapter needs to be connected to the headers on the Sonoff. The connection should be fairly obvious, with just a ground and power connector. The transmit (Tx) and receive (Rx) need to be crossed, as with an Arduino.

Simply connecting the FTDI adapter to the Sonoff is not enough for you to flash firmware or even communicate with the Sonoff. First, the Sonoff needs to be put into a programming mode:

- Wire the FTDI adapter to Sonoff, but do not plug it into the USB port.
- Press and hold the on-board button.
- Plug the FDTI into the computer and wait two or three seconds.
- Release the on-board button.

If your device doesn't have an on-board button, you can also connect GPIO 0 to ground, but in practice, I haven't heard of this being necessary with the S20.

Once the Sonoff is in flash mode, you can run your tool to flash the firmware:

```
$ sudo ./esptool.py write_flash --flash_mode dout --flash_size 1MB 0x0 sonoff.6.4.1.bin
esptool.py v2.6
Found 1 serial ports
Serial port /dev/ttyUSB0
Connecting...
Detecting chip type... ESP8266
Chip is ESP8266EX
Features: WiFi
MAC: 84:f3:eb:b0:6e:07
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Compressed 534032 bytes to 365310...
Wrote 534032 bytes (365310 compressed) at 0x00000000 in 32.4 seconds (effective 131.7 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin?
```

Note that either backing up or flashing the device will bring it out of flash mode. Moreover, the recent Sonoff switches have been shipped with a flash chip (PN25F08B) that requires flashing with ESPTool in DOUT mode.

ESPTool [4] can do more than just flash a device. It can also be used to interrogate information from the device, such as the size of the flash memory, or it can be used to back up the firmware from the device.

in a similar manner by putting the device into a special mode that causes it to become an access point with a web server. You then connect to the access point and enter the details of your WiFi configuration, which are then subsequently used to connect to your network. Flashing the S20 with Tasmota will cause it to start up as an access point automatically the first time you power up. Now, you need to connect your laptop to the new WiFi network. In my case, the network was sonoff-3591.

Once you are connected to the access point network, you must open <http://192.168.4.1> in your web browser. You will see a form similar to Figure 7 that allows you to enter your SSID and password for your local network. When this form opens, you have

# ORDER NOW!



Get 7 years of  
*Ubuntu User*  
ON ONE DVD!

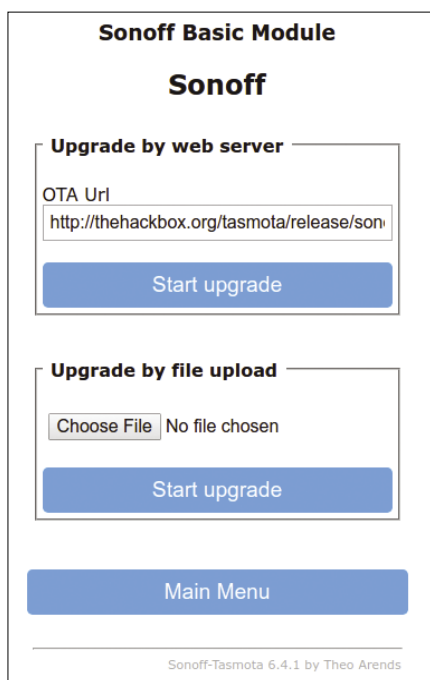


three minutes before the device automatically reboots. Pressing *Save* keeps the values and causes the device to reboot immediately.

When the device restarts, it should be reachable from your network. The device will bring up a very comprehensive menu that allows you to set up WiFi, MQTT, Domoticz, timers, and a few other settings, one of which is where you can download the most current Tasmota image when doing an OTA update (Figure 8).



**Figure 7:** Configuration dialog for WiFi.



**Figure 8:** Dialog for OTA updates.

Software, computers, and upgrades, despite rigorous testing, can occasionally encounter difficulties. Should this occur with the S20, it might be having a problem connecting to the web from the browser. If this happens, you still have a number of ways to communicate with your device. The S20 has a button that can be used for operation, such as toggling the power state; however, it can do much, much more. Most importantly, it can enable the access point. Pressing the button on the Sonoff device four times quickly causes the Sonoff to enable the access point. I couldn't find any clarification of how fast "quickly" is, but pressing the button four times in less than a second will successfully start the access point mode.

The different "commands" that can be enabled with button presses are listed in Table 1; however, if it isn't possible to connect to the Sonoff in the usual manner, this manual method might not help. Pressing the button for longer than 40 seconds will reset the settings to the defaults defined in `user_config.h` and restart the device, which might be enough to allow you to seize control without resorting to reflashing.

All of the options in Table 1 should make controlling the device easy. During my initial experiments, it wasn't possible to regain control even by reflashing. In extreme situations, you might need to erase memory before reflashing with:

```
sudo esptool.py erase_flash
```

Erasing the memory will write a 0xFF to each memory location. After the memory has been erased, it needs to be reflashed with the firmware. Make sure when you do this that you use the correct memory size for your device. Once the Sonoff is up and running, you can finally control it.

**Sending Commands**

The Tasmota firmware has a great deal of functionality to support a myriad of ESP8266-powered devices and their associated timers and sensors. Table 2 is the tiniest subset of commands from the Tasmota command list [5].

The Tasmota can be controlled by commands sent either via an MQTT broker or by sending a command to the device directly as an HTTP request. In the future, I might set up an MQTT broker or a home automation package such as Domoticz or OpenHAB, but for now, I control the devices with HTTP commands:

```
http://sonoff/cm?cmd=Power
http://sonoff/cm?cmd=Power%20On
http://sonoff/cm?cmd=Power%20Off
http://sonoff/cm?cmd=Power%20Toggle
```

It is even possible to set a password for the Tasmota web interface, which requires you to provide the username and password as part of the command:

**Table 1:** S20 Button Press Functions

Button Presses	Function
1	Toggles Relay1 directly or sends an MQTT power-on message if MQTT is configured. Blinks the LED twice and sends an MQTT status message.
2	Toggles Relay2 directly or sends an MQTT power-on message if MQTT is configured. Blinks the LED twice and sends an MQTT status message. Relay2 is supported on devices with two relays.
3	Starts the WiFi with the ESP8266 SmartConfig app, allowing for SSID and password configuration from an Android mobile phone. The LED blinks during the config period.
4	Starts the WiFi manager, providing an access point with IP address 192.168.4.1. A web server allows the WiFi configuration. The LED blinks during the config period.
5	Starts the WiFi Protected Setup (WPS), allowing for SSID and password configuration with the router's WPS button or from a web page. The LED blinks during the config period.
6	Restarts the module.
7	Starts OTA download of firmware from the preconfigured URL. The green LED lights up during the update.

**Table 2: Subset of Tasmota Commands**

Command	Payload	Description
Power<n>		Show the current power state of the relay.
Power<n>	0/off	Turn relay or Relay<n> to off.
Power<n>	1/on	Turn relay or Relay<n> to on.
Power<n>	2/toggle	Toggle relay or Relay<n> to opposite state.

```
http://192.168.178.50/cm?
user=admin&
password=passwordgoeshere&
cmd=Power%20on
```

```
curl http://192.168.178.59/cm?
cmd=Power%20Toggle
```

The command to toggle the power switch also could be added to a shell script.

Adding a password might make your devices more secure to naive attackers on your network, but it is passed in clear text, undermining your total security, especially if you access your devices over the Internet.

Controlling the Sonoff from the browser is quite neat during development, but the most flexible way would be to call a script. Although quite a few different ways of doing this probably exist, one easy example uses cURL. A number of sites discuss the full powers of cURL, but I only need to perform a simple HTTP GET:

**Conclusion**

Tasmota not only supports all the same types of functionality that most other smart socket apps support, but much, much more. Besides MQTT, it supports Domoticz home automation out of the box; more importantly, Tasmota lets you keep all your data within your home network and control the device by script or in a browser. This isn't to say I didn't encounter any difficulties during the process (see the "Troubleshooting" box), but in general I would say flashing Tasmota to my Sonoff S20 has been wildly successful. ■■■

**Troubleshooting**

Most of the problems I experienced stemmed from badly soldered header pins and an old version of ESPTool. During my journey of discovery, I also noted issues others encountered.

**Problem**  
Regardless of what commands were sent to ESPTool, the executable didn't seem to be working but didn't return any errors.

**Solution**  
The ESPTool binary from the repository might be old – or really old. Download and use the Python script [4].

**Problem**  
ESPTool was current, but it could not communicate with the device.

**Solutions**

- Bad header pin connections. I prepared two identical devices with header pins but was initially only able to communicate with one of the devices because of bad connections.
- Cheap serial adapters. Some people complained about low-quality serial adapters not working. These adapters contained FTDI chips from unverified sources. The FTDI chips might have been counterfeit. Window drivers from the providers FTDI and Prolific have been released that do not support these counterfeit chips, although this does not seem to be a problem on Linux [6] [7].
- Power. The problem could be not enough power provided to the unit. Some users have complained that adding an extension cable caused enough power loss that the FTDI adapter could not power the transfer.
- Voltage. The FTDI adapter is 5.0V, not 3.3V, and fried the device.

**Problem**  
Don't know whether the FTDI adapter is recognized by Linux.

**Solution**  
This problem can be verified with the lsusb command:

```
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 004: ID 1bcf:2883 Sunplus Innovation Technology Inc.
Bus 001 Device 003: ID 8087:07da Intel Corp.
Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub

Bus 003 Device 008: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-Serial (UART) IC

Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

The second line from the bottom shows that the device was recognized.

**Problem**  
An HTTP call to the S20 device returned strange errors, and the Sonoff didn't obey the command:

```
File Not Found
URI: /cm/
Method: GET
Arguments: 1
cmd: Power 0
```

**Solution**  
Perhaps the password was accidentally set. When passing in the command, also pass in the username and password.

**Info**

- [1] Tasmota project: <https://github.com/arendst/Sonoff-Tasmota>
- [2] Sonoff S20 schematic: [https://www.itead.cc/wiki/S20\\_Smart\\_Socket](https://www.itead.cc/wiki/S20_Smart_Socket)
- [3] Tasmota on GitHub: <https://github.com/arendst/Sonoff-Tasmota/releases>
- [4] ESPTool on GitHub: <https://github.com/espressif/esptool/releases>
- [5] Tasmota commands: <https://github.com/arendst/Sonoff-Tasmota/wiki/Commands>
- [6] Counterfeit FTDI chips: <https://vilimpoc.org/blog/2016/05/04/esptool-usb-serial-adapter-shootout/>
- [7] Changing the FTDI PID: [www.rei-labs.net/changing-ftdi-pid/](http://www.rei-labs.net/changing-ftdi-pid/)

**Author**

**Christopher Dock** is a senior consultant at T-Systems On Site Services. When he is not working on integration projects, he likes to experiment with Raspberry Pi solutions and other electronics projects. You can read more of his work on his blog <http://blog.paranoidprofessor.com>.



# MakerSpace

Wireless control over a long distance  
with the LoRa modem

## Long Range



WiFi is convenient for devices that are in the same house. If you want to extend the distance, give LoRa a call.

By Andrew Malcolm

**T**ell LoRa I love her... – Ricky Valence

Some variants of the Raspberry Pi come equipped with WiFi, which make them suitable for wireless data gathering and control applications in certain circumstances. However, the range of WiFi is limited, and it generally requires a WiFi-enabled router issuing IP addresses. Beyond WiFi, a number of standards exist for long range wireless communications in the 868MHz and 915MHz license-free bands that are more suitable for embedded applications. One of these license-free bands is LoRa [1], which is widely used and supported by several silicon implementations. The silicon itself is usually packaged onto modules, which are available for sale in the \$20 region. These modules are often further mounted on standard PCBs and offered as development kits by the silicon manufacturers and others. This article describes how to build LoRa into a Raspberry-Pi-based telemetry system and outlines the design of a low-cost HAT for the Pi with an integrated PCB antenna (which you can build for less than \$10).

### Introducing LoRa

The name LoRa is derived from “Long Range,” and long range is indeed one of the system’s defining characteristics. One group of users claims to have

achieved a 476-mile range with a transmit power of just 25mW [2]. LoRa is also low power. In the license-free ISM bands where LoRa operates [3] [4], maximum power is limited by law. But low power is often also mandated by the need for battery operation or dependence on solar energy at a remote location. So it is this combination of low power and long range that makes LoRa attractive for telemetry applications. Extreme range figures like 476 miles should be examined with a critical eye. These results are typically achieved at high altitudes with line of sight between transmitter and receiver and other optimum conditions. However, in an urban environment, ranges of up to a mile are possible with low cost, compact hardware.

All this performance comes at a price of course, as in general, the laws of physics limit range based on transmitted power, distance, signal-to-noise ratio at the receiver, and bandwidth. The more bandwidth, the more data you can transmit in a given time, and bandwidth is a limited resource. The license-free channel most used in the USA is the 915MHz band, with a bandwidth of 26MHz. This bandwidth is divided into channels, and LoRa can be configured to use channel widths of between 125kHz and 500kHz. LoRa uses spread-spectrum techniques to improve noise immunity (and thus signal-to-noise ratio), and, as a result, those longest range setups using a

125kHz channel width may be limited to data rates of a few bits per second, once packet synchronization, channel usage limits, addressing, and error detection are factored in. (Further information on LoRa modulation is available online [5].) These rates are no good for streaming live audio but are perfect for background data gathering and control. After all, to control a relay state, you simply need to

deliver one bit. Similarly, the water level in a tank or reservoir, for example to an accuracy of 1 percent, can easily be expressed in an 8-bit packet. See Wikipedia for some more interesting LoRa application examples [6].

### First Steps

To get started with LoRa, I purchased a pair of complete modules manufactured

by Semtech. This company holds patents on the LoRa standard, manufactures silicon, and sells development PCBs, as well as accessories. However, Semtech licenses to members of the LoRa alliance, so you will find plenty of alternatives. The module I chose was the SX1272RF1BAS [7], which is available from many distributors. The module is equipped with a 16-pin and a 10-pin header (Figure 1). The modem chip is controlled over a four-wire SPI interface, which is supported by the Raspberry Pi, and it is very easy to connect this module without requiring a soldering iron! The downside is that the modules cost about three times the price of the Raspberry Pi, and, of course, they come with no control or other I/O capabilities. I used some jumper wires designed to push onto the Pi's pins to connect to the modules, using the diagram in Figure 2, which was derived from the datasheets for the Raspberry Pi and the SX1272RF1BAS module.

With two Raspberry Pis connected to two modules, as well as antennas connected and powered up, it is time to fire up some software. Martin Giess's excellent SX1272 test suite [8] is a great place to start. Log onto each Pi and run the following command to download the test suite:

```
git clone https://github.com/mngiess/lora-sx1272-test.git
```

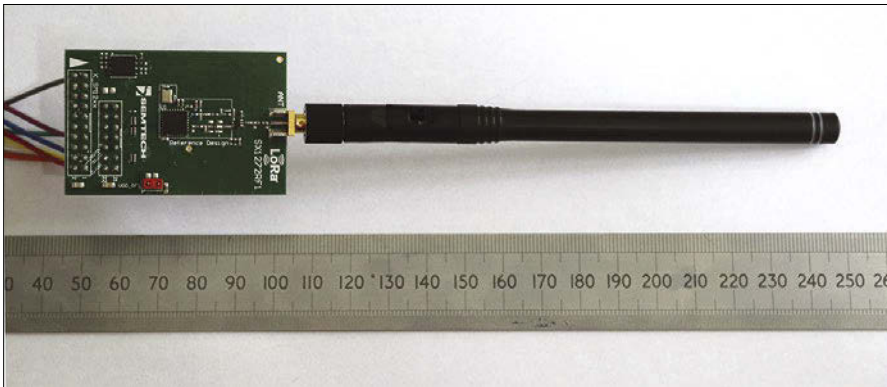
Follow the excellent instructions for reference. (Make sure you have the relevant build tools installed for your chosen distro – see the box entitled “Build Tools.”) You should end up with a transmitter and receiver application on each Pi. This software expects to find the SX1272RF1BAS module on SPI bus 0, with CS0, and that's how the modules are wired.

Run the transmitter on one Pi and the receiver on the other. If all goes well, you should see messages on the console

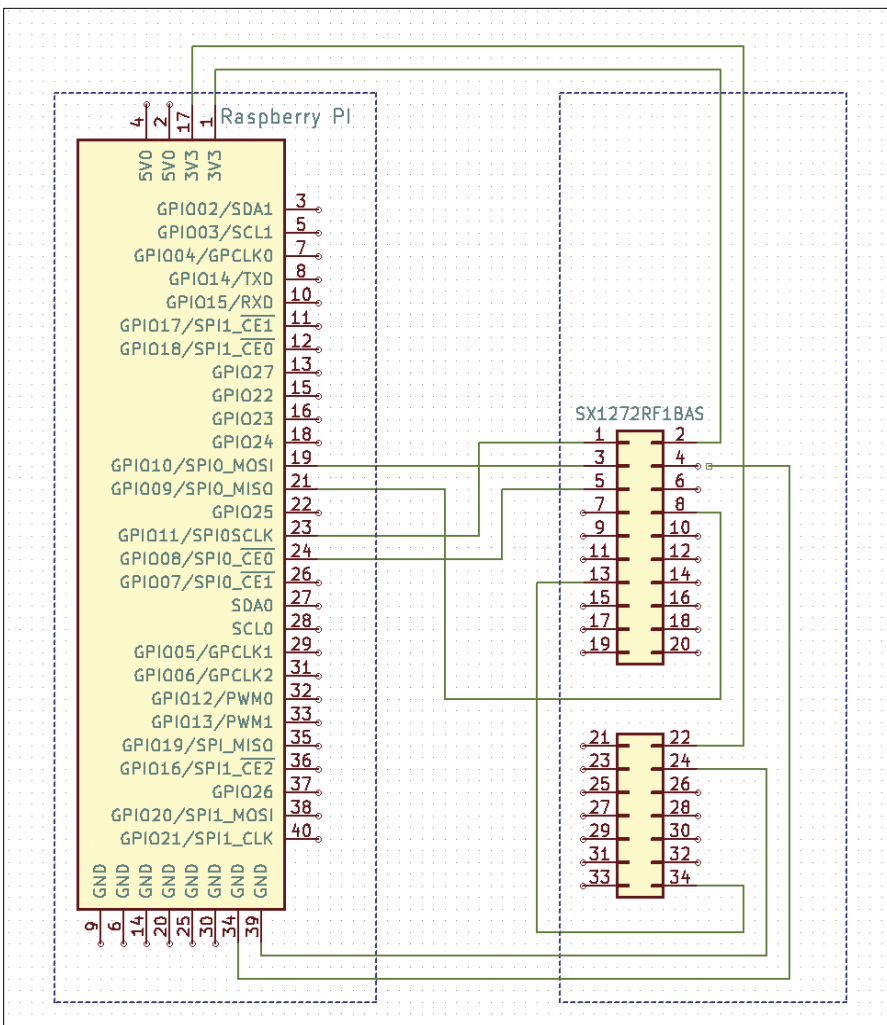
### Build Tools

On Linux systems with apt as the repo tool, installing the required build tools is as simple as:

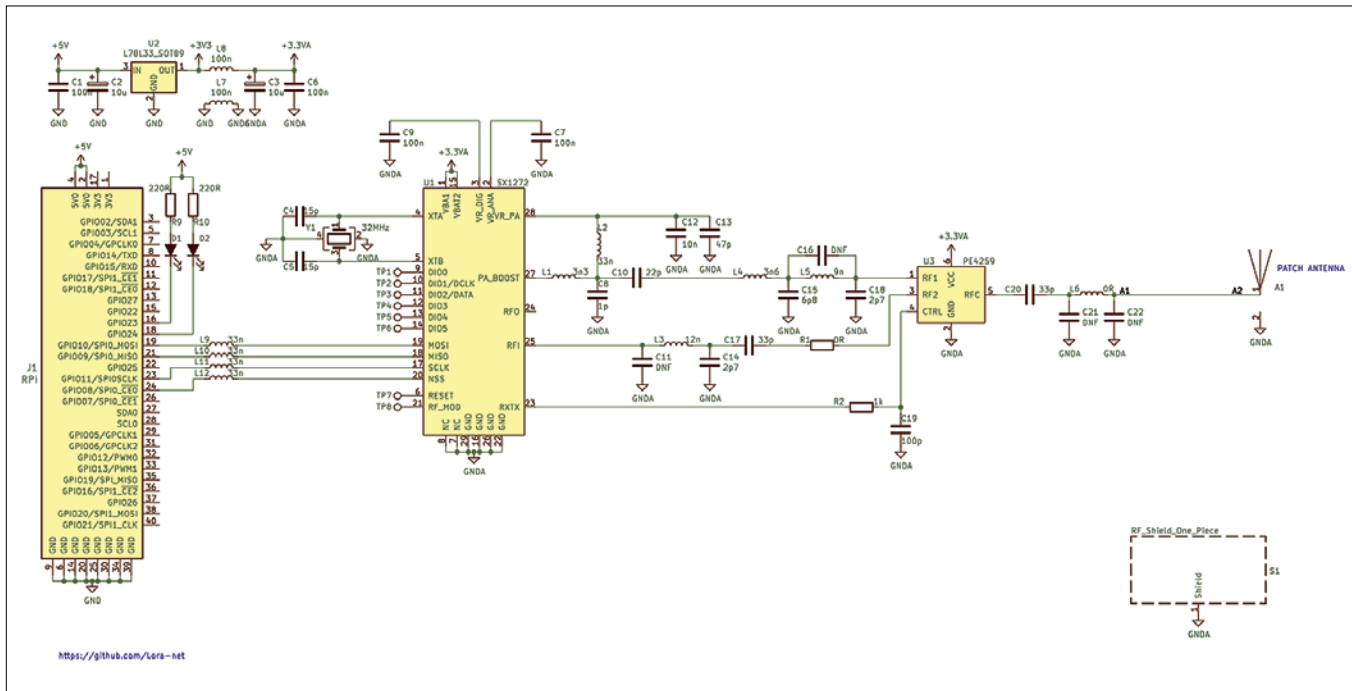
```
sudo apt-get install build-essential
On Arch Linux systems, enter:
sudo pacman -Sy base-devel
```



**Figure 1:** Semtech's module with antenna connected.



**Figure 2:** Connecting a Raspberry Pi to Semtech's SX1272RF1BAS LoRa module.



**Figure 3:** The complete LoRa HAT schematic.

of each Pi showing messages being sent by the transmitter and acknowledged by the receiver. That's it. You're up and running with LoRa! Don't forget, because these programs access hardware, they have to run as root.

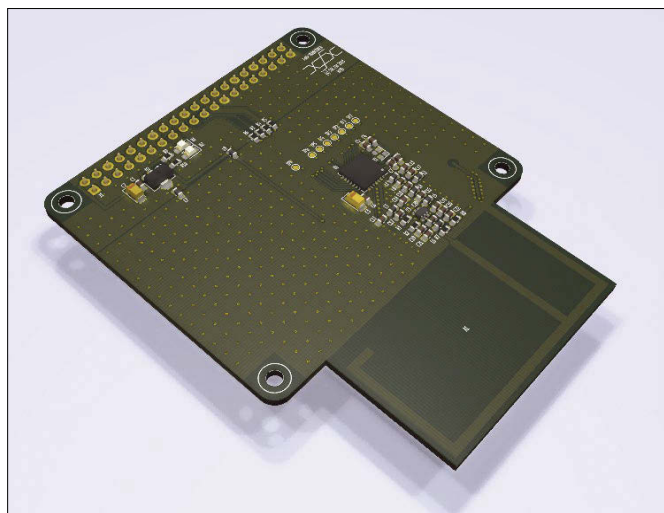
**Hardware Design**

The next step is to create a plugin or HAT for the Raspberry Pi. An initial search of the distributors turned up a number of LoRa modules in the price range of \$15 to \$20. These modules include the SX1272 (or similar) chip, an RF switch (for switching the antenna from RX to TX), and a few passive

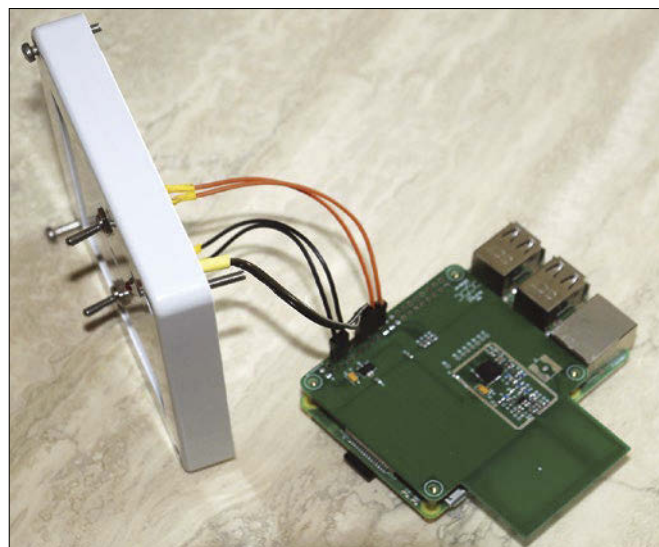
components. It is not uncommon to find these modules, which are around the size of a large postage stamp, soldered onto a larger PCB to form part of a larger system. But the SX1272 and the antenna switch (PE4259) are available for less than \$5. I needed to design a PCB anyway to interface to the Pi, so the module solution seemed overpriced. Certainly, it removes a lot of the uncertainty from the design of the RF part (from the chip to the antenna), but Semtech's website is full of resources, including complete PCB layouts for many of their modules and a lot of technical discussion on good RF

layout practice and antenna matching. On top of that, they provide a reference design for a PCB-based antenna.

The resulting compact HAT design has the antenna protruding over the HDMI connector on the Pi, and the RF circuitry itself takes up less than half the HAT PCB area, leaving plenty of space for application-specific circuitry, such as relays or sensor interfaces. This arrangement has the advantage that the whole system can be placed inside a waterproof box for deployment outdoors with no concerns over weather-proofing of the antenna or its cables. It is also much more robust, an advantage

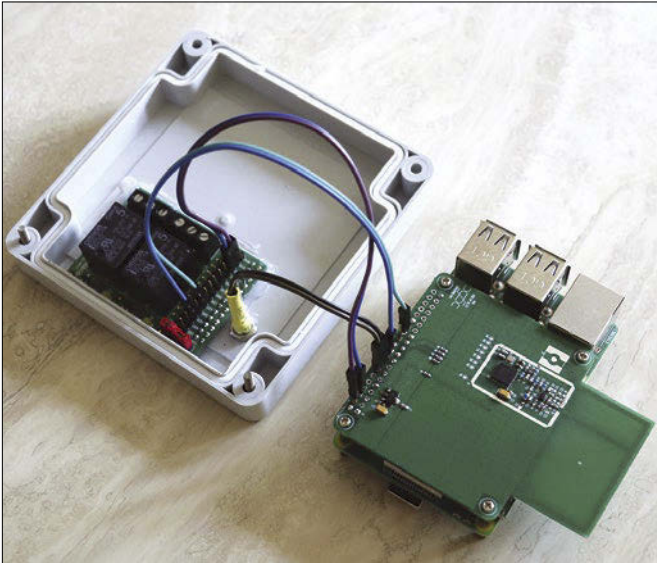


**Figure 4:** The completed design, as rendered by KiCad.



**Figure 5:** The switch module.





**Figure 6:** The relay module.

in mobile applications. In terms of I/O, the current design simply has two LEDs, as I intended to use external switches and relays in my demonstrator system. I chose to have a local 3.3V power supply derived from the Pi's 5V supply for the modem chip to try to avoid any digital noise from the Pi interfering with the RF signals. The design was implemented using KiCad, an excellent open-source schematic capture and PCB design package that runs on Linux.

The PCB layout is very straightforward (Figure 3), sticking as closely as possible to the RF layout from the SX1272 reference design [9] and the antenna reference design [10]. I increased the physical size of the passive

components from 0402 to 0603 to help ease hand-soldering. As I said earlier, the design leaves room for plenty of other circuitry, so it would be easy to modify the existing design to add application-specific circuitry. The one caveat is that the ground plane on the underside of the board forms an integral part of the antenna, so it is a good idea not to pierce it with too many long tracks. The resulting PCB can be assembled by (steady) hand in about half an hour (Figure 4).

## A Practical Telemetry System

To demonstrate the utility of the LoRa modem module, I packaged two Raspberry Pis into waterproof plastic boxes.

One Pi was set up as the transmitter running the *switch* application, with two control switches and an LED mounted externally (Figure 5). Toggling either switch causes a packet to be sent to the receiver. The LED flashes when the transmitter receives an ACK from the receiver.

The second Pi runs the *relay* application and has a small module with two relays [11] wired and mounted internally, and a single LED mounted externally that flashes on reception of a valid packet from the transmitter (Figure 6).

This is pretty much the simplest application of LoRa. You can configure a lot more, such as an example spreading factor, channel frequency, and bandwidth that allow you to tailor the radio link to a



**Figure 7:** Switch and relay modules powered from solar-charged battery packs, ready for deployment.

### Listing 1: The Switch

```

01 #include "SX1272.h"
02 #include <stdio.h>
03 #include <syslog.h>
04
05 int main ()
06 {
07     // set up the LoRa modem
08     sx1272.ON();
09     sx1272.setMode(4);
10     sx1272.setHeaderON();
11     sx1272.setChannel(CH_17_868);
12     sx1272.setCRC_ON();
13     sx1272.setPower('M');
14     sx1272.setNodeAddress(3);
15
16     // set up GPIO: 2 inputs and one output
17     bcm2835_gpio_fsel
18         (RPI_V2_GPIO_P1_18, BCM2835_GPIO_FSEL_INPT);
19     bcm2835_gpio_set_pud
20         (RPI_V2_GPIO_P1_18, BCM2835_GPIO_PUD_UP);
21     bcm2835_gpio_fsel(RPI_V2_GPIO_P1_08,
22         BCM2835_GPIO_FSEL_INPT);
23     bcm2835_gpio_set_pud(RPI_V2_GPIO_P1_08,
24         BCM2835_GPIO_PUD_UP);
25     bcm2835_gpio_fsel
26         (RPI_V2_GPIO_P1_16, BCM2835_GPIO_FSEL_OUTP);
27
28     // get the current switch states
29     uint8_t oldLev_08=bcm2835_gpio_lev
30         (RPI_V2_GPIO_P1_08);
31     uint8_t oldLev_18=bcm2835_gpio_lev
32         (RPI_V2_GPIO_P1_18);
33
34     while(1)
35     {

```

**Listing 1: The Switch (continued)**

```

29         // get the new states
30         uint8_t lev_08=bcm2835_gpio_lev
           (RPI_V2_GPIO_P1_08);
31         uint8_t lev_18=bcm2835_gpio_lev
           (RPI_V2_GPIO_P1_18);
32
33         //if state is different, send message and
           update
34         if(lev_08 != oldLev_08)
35         {
36             char *msg=(char*)
               (!lev_08?"ON0":"OFF0");
37             syslog(LOG_USER,msg);
38             uint8_t e = sx1272.
               sendPacketTimeoutACKRetries(8, msg);
39
40             if(e==0)
41             {
42                 // flash the LED
43                 bcm2835_gpio_set
                   (RPI_V2_GPIO_P1_16);
44                 bcm2835_delay(500);
45                 bcm2835_gpio_clr
                   (RPI_V2_GPIO_P1_16);
46                 // update rthe state
47                 oldLev_08=lev_08;
48             }
49         }
50
51         //if state is different, send message and
           update
52         if(lev_18 != oldLev_18)
53         {
54             char *msg = (char*)
               (!lev_18?"ON1":"OFF1");
55             syslog(LOG_USER,msg);
56             uint8_t e=sx1272.
               sendPacketTimeoutACKRetries(8, msg);
57
58             if(e==0)
59             {
60                 // flash the LED
61                 bcm2835_gpio_set
                   (RPI_V2_GPIO_P1_16);
62                 bcm2835_delay(500);
63                 bcm2835_gpio_clr
                   (RPI_V2_GPIO_P1_16);
64                 // update rthe state
65                 oldLev_18=lev_18;
66             }
67         }
68
69         // wait a bit & repeat
70         bcm2835_delay(500);
71     }
72 }

```

specific situation. I was able to outfit both modules to run on solar-charged battery packs (Figure 7).

**Software**

The software for this system builds on the test software described earlier. The complete code for both switch and relay applications, together with a makefile are available for download from my GitHub page [12]. Only the end-user parts of the code are listed here; for brevity, I have omitted error-reporting code.

Listing 1 shows the switch module. This module sets up the required GPIO pins and then enters a loop looking for pin changes from the external switches. In detecting a change, it sends a message to the receiver to turn on or off the associated relay.

Listing 2 shows the relay module, which sets up the required GPIO pins and then enters a loop listening for messages from the switch module. In detecting a message, it turns on or off the associated relay.

**Next Steps**

For further development, I should point out that LoRa data flow is truly bidirectional. There's nothing to stop the remote node, in this case the relay node, from sending unsolicited packets to the switch node. So any node can be both a sender and a receiver. It is also possible to send broadcast packets to any node listening on the same channel, as well as to send unacknowledged packets, similar to TCP/IP's UDP packets. A complete, freely available WAN implementation called LoRaWAN [13] even turns a bunch of LoRa nodes and a gateway node into a true network, with semantics very similar to TCP/IP.

**Conclusion**

I hope the system described in this article, and the details of how it was built, will prove of interest. For an in-depth understanding of the modem chip, see the datasheet [14], as well as many of the LoRa resources on Semtech's site. There is always scope for improvement, both in hardware and software. The LoRa HAT board could be put to many

different uses with simple changes and additions to the software. The SX1272 software module provides a very simple abstraction of the modem function, so replacing something like an RS485 link or an existing TCP/IP messaging system would be fairly simple. For a more in-depth understanding of LoRa and its applications, see the many excellent YouTube tutorials on the subject – and on low-power radio in general. ■■■

**Info**

- [1] LoRa: <https://en.wikipedia.org/wiki/LoRa>
- [2] LoRa world distance record: <https://www.thethingsnetwork.org/article/lorawan-distance-world-record>
- [3] US 915MHz licence-free band: [https://en.wikipedia.org/wiki/ISM\\_band](https://en.wikipedia.org/wiki/ISM_band)
- [4] European 868MHz license-free band: [https://en.wikipedia.org/wiki/Short-range\\_device#SRD860](https://en.wikipedia.org/wiki/Short-range_device#SRD860)
- [5] LoRa modulation basics: <https://www.semtech.com/uploads/documents/an1200.22.pdf>

**Listing 2: The Relay**

```

01 #include "SX1272.h"           34
02 #include <stdio.h>           35         if(strcmp("ON0", msg)==0)
03 #include <string.h>          36         {
04                               37
05 int main ()                  37             bcm2835_gpio_set
06 {                             38             (RPI_V2_GPIO_P1_18);
07     // set up the LoRa modem  38         }
08     sx1272.ON();              39
09     sx1272.setMode(4);        40         if(strcmp("OFF0", msg)==0)
10     sx1272.setHeaderON();     41         {
11     sx1272.setChannel(CH_17_868); 42             bcm2835_gpio_clr(
12     sx1272.setCRC_ON();       42             RPI_V2_GPIO_P1_18);
13     sx1272.setPower('M');     43         }
14     sx1272.setNodeAddress(8); 44
15                               45         if(strcmp("ON1", msg)==0)
16     // set up GPIO: 3 outputs  46         {
17     bcm2835_gpio_fsel          46             {
18         (RPI_V2_GPIO_P1_26, BCM2835_GPIO_FSEL_OUTP); 47             bcm2835_gpio_set
19     bcm2835_gpio_fsel          47             (RPI_V2_GPIO_P1_26);
20         (RPI_V2_GPIO_P1_18, BCM2835_GPIO_FSEL_OUTP); 48         }
21     bcm2835_gpio_fsel          49             {
22         (RPI_V2_GPIO_P1_16, BCM2835_GPIO_FSEL_OUTP); 49         }
23                               50         if(strcmp("OFF1", msg)==0)
24                               51         {
25     // turn both relays off    51             {
26     bcm2835_gpio_clr(RPI_V2_GPIO_P1_18); 52             bcm2835_gpio_clr
27     bcm2835_gpio_clr(RPI_V2_GPIO_P1_26); 52             (RPI_V2_GPIO_P1_26);
28                               53         }
29                               54
30     while(1)                  54         {
31     {                             55             // flash the LED
32         // wait for an incoming message and sens 55             an ACK
33         // an ACK              56             bcm2835_gpio_set
34         if (sx1272.receivePacketTimeoutACK(10000) == 0) 56             (RPI_V2_GPIO_P1_16);
35         {                             57             bcm2835_delay(500);
36         // get the packet data and act on 57             it
37         // it                    58             bcm2835_gpio_clr
38         char *msg =              58             (RPI_V2_GPIO_P1_16);
39         (char *)sx1272.packet_received.data; 59         }
40         printf("%s\n", msg);     60         }
41         sx1272.getRSSI();        61     }

```

- [6] LoRa example deployments: [https://en.wikipedia.org/wiki/LoRa#Deployments\\_of\\_LoRa\\_Technology](https://en.wikipedia.org/wiki/LoRa#Deployments_of_LoRa_Technology)
- [7] Semtech's SX1272RF1BAS module: <https://www.digikey.co.uk/product-detail/en/semtech-corporation/SX1272RF1BAS/SX1272RF1BAS-ND/4490407>
- [8] LoRa test software for the Raspberry Pi: <https://github.com/mngiess/lorasx1272-test/blob/master/README.md>
- [9] SX1272 modules: SX1272RF1xAS – 868 or 915 MHz – Combined RFI and PA\_BOOST design: <https://www.semtech.com/products/wireless-rf/loras-transceivers/sx1272>

- [10] Planar F-Antenna Reference Design: [https://www.semtech.com/uploads/documents/AN1200.20-SARANT\\_V1\\_0\\_STD.pdf](https://www.semtech.com/uploads/documents/AN1200.20-SARANT_V1_0_STD.pdf)
- [11] Slice of relay – Pi accessory: <https://rlx.sk/en/raspberry-pi/2120-slice-of-relay-raspberry-pi-b047-relay-board-for-raspberry-pi-.html>
- [12] Author's GitHub project page: [https://github.com/andrewrussellmalcolm/loras\\_hat](https://github.com/andrewrussellmalcolm/loras_hat)
- [13] LoRaWAN introduction: <https://www.thethingsnetwork.org/docs/lorawan/>
- [14] SX1272 datasheet: [https://www.semtech.com/uploads/documents/SX1272\\_DS\\_V4.pdf](https://www.semtech.com/uploads/documents/SX1272_DS_V4.pdf)

**Author**

**Andrew Malcolm** (MIEE, CEng) is a software engineer for Guru Systems (<https://www.gurusystems.com/>), a fast-growing IoT hardware and SaaS company working on low carbon energy projects. In his spare time, he likes to combine software engineering with his first love, hardware engineering. With all the open source tools available, he is never short of things to design. The Raspberry Pi has proved a source of inspiration. To date, Andrew has designed five different add-ons, or HATs. He is currently working on micro-stepping motor drives for a Pi-based laser cutting machine. You can reach him at [andrewrussellmalcolm@gmail.com](mailto:andrewrussellmalcolm@gmail.com).





# MakerSpace

Designing Keyboardio

## The Making of a Keyboard

Jessie Vincent, cofounder of Keyboardio, looks back at the process of designing an efficient and comfortable keyboard and some of the design decisions and challenges involved.

By Bruce Byfield

**T**he Keyboardio [1] Model 01 is an open hardware success story (Figure 1). First shipped in late 2017, it has gone on to ship thousands of units, as well as developing a small but enthusiastic band of users and volunteer developers. However, this success did not come overnight. Besides the challenges that face a small and new manufacturer, the Model 01 also faced endless technical decisions about everything from the shape of the keyboard to the choice of LED switches and their position on the circuit board. Recently, Jesse Vincent, cofounder and CTO of Keyboardio, took the time to shed some light on the product development.

Ever since Vincent's first summer internship as a programmer, Vincent had found himself prone to repetitive stress injuries. "For quite a while," he remembers, "I carried a Microsoft Natural Elite keyboard in my backpack. I

tended to go through three or four of them each year." Sometime in 2011 or 2012, "I discovered that there was an online community of folks who made their own keyboards. Somewhat naively, I figured I'd take a month and build myself a keyboard. Seven years later, my wife Kaia and I work full-time on keyboards."

Although keyboards remain the main device for computer input, their design has changed little in the last few decades. Although alternate keyboard layouts are available, the majority of keyboards continue to use the QWERTY layout, so-called from the left-hand top bank of keys.



**Figure 1:** Keyboardio's Model 01 has become an open hardware success story.

Lead Image © 3955m, 123RF.com

In fact, Vincent observes, “most traditional keyboards are based on a typewriter design by Christopher Sholes from the 1880s. While there are a number of widely repeated anecdotes, we don’t know a whole lot about the reasoning behind his design. It certainly wasn’t based on how human bodies work or what’s comfortable or productive. Touch-typing wasn’t even invented until Sholes’ QWERTY typewriter was already in regular use. My favorite, possibly-apocryphal story about the Sholes QWERTY layout is that it was designed so that a salesperson could type the word ‘typewriter’ using only the top row of keys.”

In designing the Model 01, Keyboardio drew inspiration from other ergonomic keyboards, with the Model 01 making a number of standard departures from the standard keyboard. It is a minimalist keyboard, without separate function, arrow, or numerical keys, all of which are accessed through keyboard combinations. Add curved banks of keys to the minimalist keys, and the average human hand does not need to stretch so far. Similarly, keys are arranged in linear rather than staggered columns, reducing how far fingers must travel from key to key. A split keyboard whose halves could be tilted along two axes was still another standard ergonomic feature that became part of the Model 01.

In addition, like gaming keyboards, the Model 01 has mechanical switches for keys, which allow for durability and, if desired, aural feedback, as well as programmable keys and the swapping of keyboard layouts.

“We looked at dozens and dozens of typewriters and keyboards,” Vincent says. The most direct influence was the Tron TK-1 (Figure 2), which was designed for use with Ken Sakamura’s Tron operating system [2]. “It featured a columnar layout similar to the Model 01, with each column of keys at a different angle.” An added attraction was that “the TK-1’s creators designed the physical layout of the keyboard based on a number of studies of how human hands and fingers actually work.”

To these sources, Keyboardio added its own touches, such as individually sculpted keys to help guide typing fingers along minimal paths from key to key and a palm key. However, this general set of

goals was only the beginning. A tour of over 25 hacker spaces throughout North America, as well as appearances at conferences like OSCON, gave Keyboardio feedback from over one thousand people, providing invaluable feedback as well as testing the practicalities of design (Figure 3). The devil, as Keyboardio soon discovered, was in the engineering details.

### Issues Along the Way

A chronological account of designing the Model 01 would be repetitive, with many issues being dealt with at the same time, and apparently resolved issues needing to be revisited. To avoid confusion, what

follows is a description of only some of the design issues.

To start with, according to Vincent, “we made several dozen prototypes on the way to the Model 01 (Figure 4). The earliest prototypes were laser-cut out from a single sheet of acrylic and point-to-point soldered to an Arduino, without



**Figure 2:** The Tron TK-1 was one of the inspirations for the Keyboardio Model 01.



**Figure 3:** Keyboardio's Jesse Vincent and Kaia Dekker testing the Model 01 at OSCON.



**Figure 4:** Some of the early prototypes for the Model 01.



a circuit board holding everything together. This sort of design is very, very fragile, but it's also very quick. We were often able to go from a working idea to a working keyboard within 24 hours. Most of our early prototypes were one-piece keyboards. They were mostly fairly small and portable, but lacked the adjustability that makes the Model 01 so adaptable."

Vincent continues, "The earliest looked very similar to other well-known ergonomic keyboard layouts. As we tested them, we found that they just weren't very comfortable. Over time, we moved the keys around bit by bit until

we hit on something that felt right."

Judging from photos, an ongoing problem was the positioning of control keys. For example, the prototype in the top right of Figure 4 places the Ctrl keys inefficiently beside each other. Similarly, the prototype in the bottom left has only a single key on one side of each bank of keys. Some prototypes also tested such features as the shape of the two halves of the keyboard, which result in Keyboardio's butterfly logo, and the connecting mechanism for the two halves.

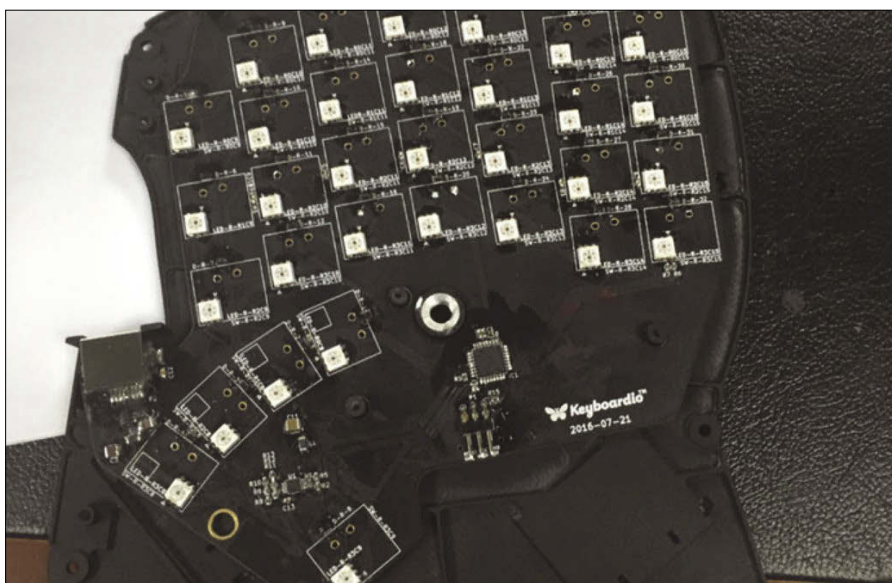
Initially, Keyboardio hoped to include Bluetooth support. Bluetooth chipsets

and batteries were priced, and drivers were written. However, the idea soon proved impractical. Each LED backlight drew as much power as a Bluetooth radio. "So we'd either have to disable the LEDs in Bluetooth mode or have a really big battery," Keyboardio explains in an early blog [3]. "A really big battery in one half of the keyboard would make one side of the keyboard heavier than the other. So we'd have to either add a weight on the other side to counterbalance the battery or a second battery charging chip on the other side. And then it turns out that if you put a battery inside an electronic product, there are countries that won't let you ship by air."

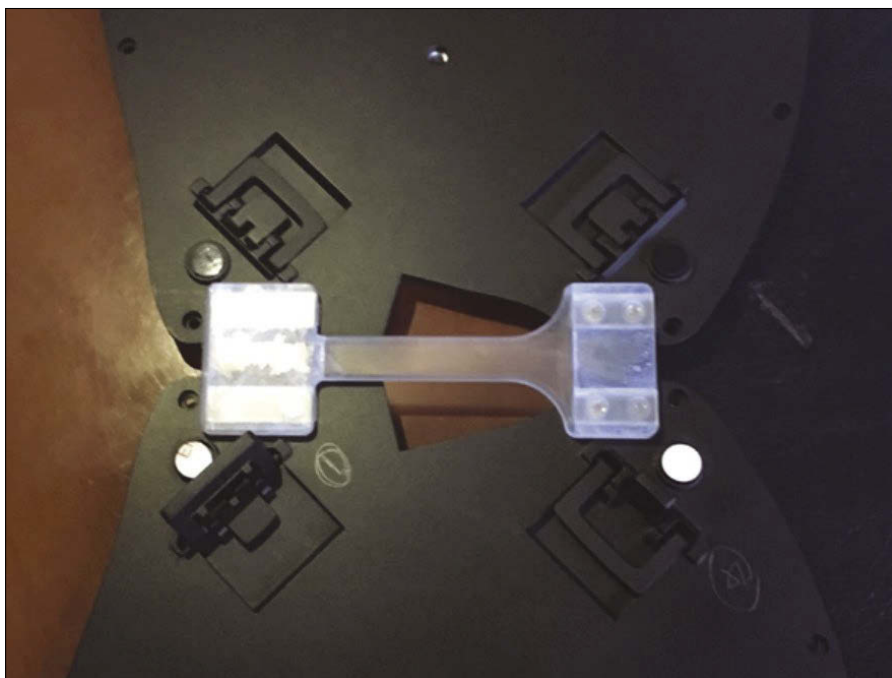
Furthermore, the blog explains, "It's hard to make the economics work once you start pricing out adding a Bluetooth chipset and a battery charger to a low-volume product like the Model 01. Most of the folks we survey who were interested in having Bluetooth weren't all that interested in having it at a price we could afford."

At another point [4], Vincent discovered a problem with new circuit boards: "I drove home, plugged the board into my firmware flasher and ... nothing. Actually, it was a little worse than nothing. I saw a tiny little puff of magic smoke rise from the board. We'd inadvertently had a diode mounted backwards on the PCB. When the programming cable was plugged in correctly, it couldn't tell the microcontroller to reboot into programming mode. When the programming cable was plugged in upside down, the diode got hit with 5 volts of power coming in the wrong direction and popped." Further investigation showed that "the right-hand PCBs were drawing insane amounts of power, not ever really powering up properly, and were getting really, really hot. A close reading of the schematics for the two boards turned up a tiny, tiny miswiring. One of the pins on the keyscanning controller was wired to ground when really, it was supposed to be wired to the 5 volt line. So it was doing something screwy inside the controller to try to get that power, but being thwarted by the external ground connection (Figure 5)."

At least twice, the LEDs that backlight the keys came to the forefront in design. The first time, Vincent switched to APA102C LEDs and put the LEDs on the



**Figure 5:** An early prototype of the Model 01's printed circuit board.



**Figure 6:** An early version of the Model 01's connecting hinge. Both the hinge and feet underwent extensive modifications during design.



same circuit board as the switches, eliminating two large PCBs from each keyboard and making everything just a bit easier to put together [5]. The blog reported that “the new LEDs are a bit brighter, a bit easier to program, and can update quicker, which means there should be less of the flickering that sometimes afflicts LEDs.”

Later on, the LEDs started to fail [6]. The blog recalls that “A little bit of triage with a magnifier revealed the problem: about 60% of the LEDs looked cloudy inside. It turned out that the PCBA shop hadn’t properly treated the reel of LEDs we’d consigned to them. Electrical components typically ship from the factory in air-tight packaging. They can absorb moisture when they’re exposed to the open air for more than a few days. As it happened, there was a nick in the air-tight foil packaging of our sample reel, so our LEDs had been slowly absorbing a bit of moisture over the course of a few months.”

These are only a few of the design issues Keyboardio faced. Others include the decision to use metric measurements and getting the feet and hinge correct so that users could angle the halves of the keyboards to fit their preferences (Figure 6). Quite late in the process, the exact shapes of some of the keys were tweaked, partly for aesthetics, and partly for problems of some keys getting caught against each other

when pressed. Even after the first Model 01s shipped, Keyboardio faced issues such as key mechanisms that required lubrication, and some feet that did not lie flat. Some problems occurred because of a tendency of factories to substitute specified parts without informing the company. Judging from posts on the Keyboardio Community forums, such problems eventually decreased in frequency, but clearly the issues came from a variety of sources ranging from design to components and were more complex than might be imagined.

### Juggling Chainsaws

In the end, the Model 01 became a success, and Keyboardio gained a reputation for transparency about its engineering and design. Even so, many features were jettisoned as impractical. According to Vincent, these features included pointing devices, wireless connectivity, and analog input – keys that type a different character when different amounts of pressure are applied, one of the newest advances in keyboard technology. Vincent declines to discuss unreleased products, but acknowledges that Keyboardio is looking at such features “really hard” for the future.

Meanwhile, thanks to Keyboardio’s detailed account of its design issues, the Model 01’s design process is a concrete example to other open hardware

developers of exactly what they can expect when they attempt to bring a product to market. With so many factors to consider at the same time, the process can be likened to the busking acts who juggle chainsaws. Yet, despite all the delays and things that can go wrong, Keyboardio proves that new entrepreneurs can achieve at least small-scale success, if only they plan and persist. ■■■

### Info

- [1] Keyboardio: <https://shop.keyboard.io/>
- [2] Tron operating system: [https://en.wikipedia.org/wiki/TRON\\_project](https://en.wikipedia.org/wiki/TRON_project)
- [3] Bluetooth problems: <https://www.kickstarter.com/projects/keyboardio/the-model-01-an-heirloom-grade-keyboard-for-seriou/posts/1273605>
- [4] Circuit board problems: <https://www.kickstarter.com/projects/keyboardio/the-model-01-an-heirloom-grade-keyboard-for-seriou/posts/1543469>
- [5] LED changes: <https://www.kickstarter.com/projects/keyboardio/the-model-01-an-heirloom-grade-keyboard-for-seriou/posts/1309005>
- [6] LEDs failing: <https://www.kickstarter.com/projects/keyboardio/the-model-01-an-heirloom-grade-keyboard-for-seriou/posts/1543469>

### Special Thanks

All pictures courtesy of Jesse Vincent.

Available Now

# \* 2018 EDITION \*

## Linux Magazine Archive DVD

**ORDER NOW!**  
shop.linuxnewmedia.com

Searchable  
Linux Archive:  
**19,000**  
Pages of Practical  
Know-How

**214 issues of Linux Magazine**  
on one handy DVD!

- ▶ All the Tricks
- ▶ All the Hacks
- ▶ All the Apps

ISSUE 215 OCT 2018

**LINUX**  
MAGAZINE

Although this Linux Magazine disc has been tested and is to the best of our knowledge free of malicious software and defects, Linux Magazine cannot be held responsible, and is not liable for any disruption, loss, or damage to data and computer systems related to the use of this disc.



**THE COMPLETE  
LINUX MAGAZINE  
ARCHIVE**

# MakerSpace

## A networked receiver for digital music Roaring Cube

Build a networked receiver for your digital music collection with an old stereo, a Raspberry Pi, and the HiFiBerry.

By Friedhelm Greis

**R**ecivers are much like cars in some respects: While the world is busy talking about networked, autonomous, and electric vehicles, cars that simply drive reliably from A to B seem to be completely antiquated. Likewise, the HiFi amplifiers still found in many living rooms might deliver outstanding sound to your speakers, but their strong suit is not playing a digital music collection. In the end, you need to network your HiFi system. In this article, I show you how to build such a system with a Raspberry Pi and HiFiBerry.

In my case, I started out with a Philips FW362 [1] compact system built in 1999, with a twin cassette deck and a CD player (Figure 1). Even at the time of purchase, the stereo system was definitely not for audiophiles. The CD player died years ago, but the cassette deck now radiates retro charm.

Nevertheless, expanding the system by simply adding a network player is not the best idea; rather, I want to integrate a networked receiver with an integrated power amplifier. You can find a quite large selection of both device types today. The question therefore arises: What features do I need, and which devices offer them?

This question is not easy to answer for many products because of the numerous features and standards on

offer: WiFi, Ethernet, Bluetooth, USB, DAB (Digital Audio Broadcasting), DLNA (Digital Living Network Alliance), FlareConnect, DTS Play-Fi, MusicCast, Dolby Atmos, Spotify, the HEOS App, and so on. You might think you are equipped for any digital case, but only at first glance (see the “Commercial Receivers” box).

### Requirements

For my purposes, the receiver needs to meet at least the following requirements: It has to play my music collection from the home network-attached storage (NAS), recognize files on USB media, and transmit sound to a Bluetooth headset. Additionally, I want to be able to operate all the basic functions without an additional device, such as a smartphone app. For example, navigation of the integrated displays and buttons must be so convenient that titles on the NAS can be found and played. The icing on the cake would be a Wake-on-LAN (WoL) function for the NAS.

The rationale behind these requirements is that audio receivers are a long-term purchase that should normally continue to function reliably after several years. Whether they will still be able to communicate with the latest devices in 10, 15, or even 20 years is another matter. If not, they would be electronic scrap again, even if they are still



**Figure 1:** If I’m honest, this Philips FW 362 compact system from 1999 with a defective CD player is electronic scrap.

### Author

**Friedhelm Greis** ([fg@golem.de](mailto:fg@golem.de)) is editor for network politics at Golem.de. He studied electrical engineering, theology, Spanish, philosophy, and journalism in Trier and Mainz (Germany) and in Bolivia. He runs the Tucholsky [16] blog [Sudelblog.de](http://Sudelblog.de) and writes for Wikipedia.



working flawlessly. I find additional Bluetooth adapters that plug into the headphone jack cumbersome to use, and they constantly need to be plugged and unplugged. Pairing can also be very unreliable.

### HiFiBerry

Buying a brand new receiver feels somewhat like buying an electric car. You can spend a large amount of money without even getting close to the range of the cheapest gasoline model. The alternative

to a complete receiver is an independent network player. You can connect it to an existing system and then, for example, play music to the amplifier through a Bluetooth adapter. Other network players can in turn be integrated into the

### Commercial Receivers

With the first test device, an Onkyo R-N855, I noticed that the first impression can be misleading (Figure 2). To its advantage, the R-N855 does not necessarily require an app for operation. All you need is the remote control and the buttons on the device. The network connection can be configured in a web front end. However, the use of WiFi and local area network (LAN) together are limited: If you disconnect the network cable, you have to enable the WiFi connection manually again.



**Figure 2:** The Onkyo R-N855 is a typical network receiver with only a few front panel controls.

The device has only a few controls. The small LCD lets you scroll through the NAS folders for music tracks, but control is very limited. Only after a firmware update could I fast forward through titles, although I could not stop a track with the fiddly menu button. Only the remote control or a smartphone offer really useful control.

Not surprisingly, the few switches and buttons and the display are obviously not designed to operate the receiver. Instead, a smartphone can be connected to the R-N855 by Bluetooth to stream music to the device from various apps such as Spotify.

When connecting an external solid-state drive with the USB port, though, I noticed that the R-N855 failed to identify the drive. The reason is given in the operating instructions: “The unit is also compatible with USB storage devices using the FAT16 or FAT32 file system formats. Other formats such as exFAT, NTFS, and HFS cannot be played by this unit” [2]. Because my hard drive was ex-FAT-formatted, the receiver could not read it.

The device has its own Bluetooth button on the front, but you can only connect Bluetooth devices that send their data to the receiver. Outputting the data to a Bluetooth headset or earplug does not work. Moreover WoL is only a pipe dream, which means the Onkyo R-N855 is out of the running as a network receiver for my purposes, despite a price of just under \$800 (£500, EUR455).

I repeated this process with other test devices. The Denon AVR-X1400H (\$450/£500/EUR300) has basically the same functions with the same weaknesses. The receiver can only play FAT16 and FAT32, and output via Bluetooth is again impossible. The

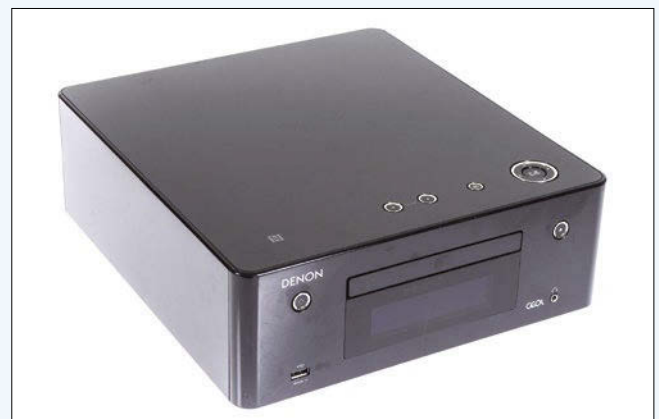
device has a few more buttons than the Onkyo receiver, but using them is finicky, so you cannot start or stop tracks. The receiver is primarily designed to produce a cinematic sound in your living room with its 7.1-channel surround system. Additionally, the device can play music on various loudspeakers in the



**Figure 3:** The Denon AVR-X1400H has more buttons than the Onkyo R-N855, but they hardly make operation any easier.

home with the HEOS WiFi multiroom sound system (Figure 3).

What seemed best suited to my purposes was a compact system like the Denon CEOL-N10 (\$400/£360/EUR400). It has a slightly larger display and a CD player (Figure 4). Even Alexa voice control is possible. Like the AVR-X1400H, the CEOL-N10 can distribute music to several rooms through the HEOS system. Various music services can also be streamed to the system from an app, but almost the entire industry seems to have agreed not to support Bluetooth headphones with their receivers.



**Figure 4:** The Denon CEOL is a compact system with a CD player.

One exception is the Yamaha RX-A670 receiver (now discontinued, but available used), which has all imaginable features and even an integrated Bluetooth audio transmitter, as does the somewhat older RX-V681. However, the newer R-N803D, also with good connectivity but at even more expense, sees Yamaha drop this feature. Again, though, all the Yamaha devices only support FAT16 and FAT32.





**Figure 5:** A seven-inch touch display and the Kodi media center make it relatively easy to browse a music collection.



**Figure 6:** The Rasp Pi comes with at least four USB ports. It can be networked over the network drive and WiFi.

home network via DLNA. One good example of this is the Yamaha NP-S303. It can play music on headphones and speakers via Bluetooth and is otherwise very well networked, but without remote control and an app, nothing happens. The two-line LCD display supports very limited use without the MusicCast app.

In a test, the German Frankfurter Digital Newspaper [3] pointed out another way to network your stereo system with the help of a Raspberry Pi computer and a plug-in sound card like the HiFiBerry [4]. Even in terms of sound quality, this combination is said to keep up with the easily more expensive test devices.

HiFiBerry offers a small box for combining with the Raspberry Pi, but I am looking for a standalone model that can be operated without an app, which would require a touch-capable display and thus a larger housing.

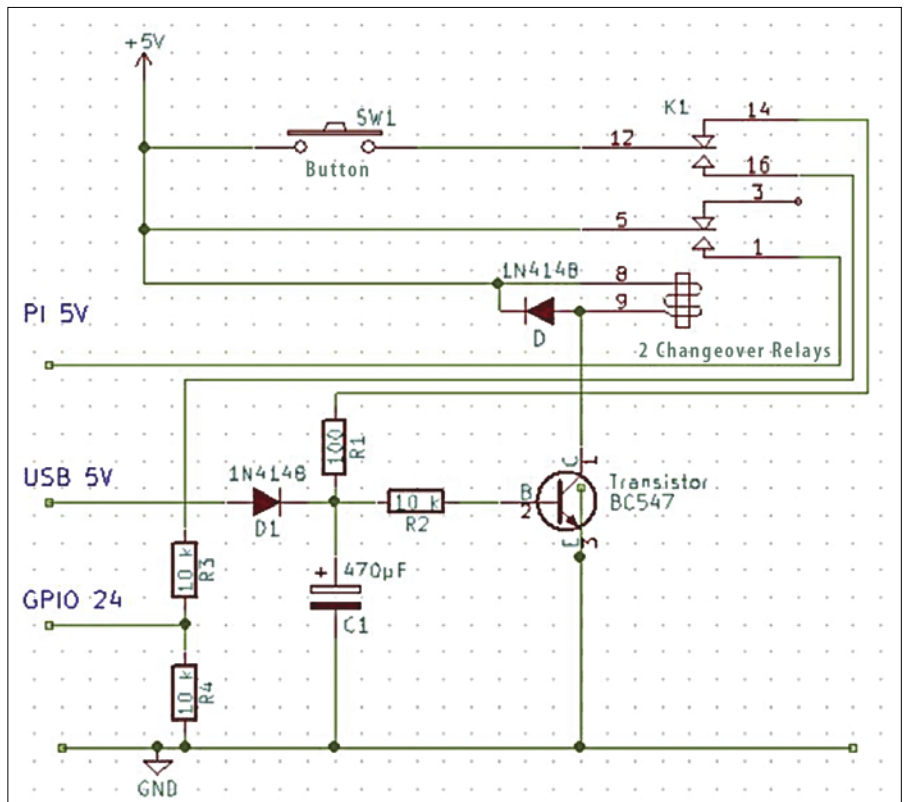
Because my Philips CD player is broken, a network drive removes the need for CDs anyway, and the player sits right at the top of the case, I had no problem freeing up some space for the display (Figure 5). More than enough space is still available below the plastic hood to accommodate the Rasp Pi and HiFiBerry. If both boards are positioned correctly in the corner, all outputs can be used. The thin plastic also makes it easy to cut out the recesses for the display and connections (Figure 6).

### On/Off Switch

Because the Raspberry Pi does not have an on/off switch, I will be adding another board with that function to the two

boards already in place (Figure 7). The now unused button that previously operated the drawer of the CD player can be used for my purposes. You can find a number of how-tos on the web for on/off switches, some of which need a complete microcontroller. However, I designed a circuit with a simple resistor-capacitor (RC) circuit and a transistor-controlled self-holding relay, which allows me to power on and boot the Rasp Pi (Figure 8).

Additionally, the same drawer button can be used to call several functions, which I query with a Rasp Pi GPIO input (Figure 9). Depending on the number of operations, the Rasp Pi can then be shut down or rebooted with a small Python script. The Raspberry Pi's USB output, unlike the 5V GPIO pins, is switched off at shutdown. The capacitor of the RC circuit then discharges, and the self-holding relay interrupts the power supply again.



**Figure 7:** The circuit diagram for the on/off switch. The former eject button of the CD drawer serves as a button.

## Sound Card

At first, my plan was to connect the output of the HiFiBerry to the AUX input of the system. I wanted to switch the two inputs with a push-button switch, but by chance, I came across a circuit diagram for the identical Grundig 18-C compact system [5] and discovered which pins of the CD player were attached to the sound fader control circuit (SOFAC). I was therefore able to connect the HiFiBerry output directly through the corresponding connector on the board. A shielded network cable eliminated hum (Figure 9).

To avoid having to connect the RCA outputs, the HiFiBerry board already has holes for a post connector. The warranty expires if you solder on the attachment, but this can hardly be avoided if you still want to use GPIO pins. They can only be reached through the HiFiBerry because of the attachment. Please note that certain pins are already reserved [6]. Power is also supplied by the HiFiBerry.

## LibreELEC

The question arises as to which software to use on the receiver. Volumio, for example, can be used to stream music [7]. In my case, however, my compact system uses one out-of-date Euronorm standard [8], and my Samsung television uses an equally ancient Euronorm standard. The TV is already smart and networked, but it has had trouble playing back certain content for some time now. As early as 2017, Samsung removed the YouTube app [9];

moreover, it has no usable web browser to play back video offerings from the web.

I therefore chose a dual-boot system for the free LibreELEC Linux operating system based on the Kodi media center. For example, BerryBoot and OpenELEC or Noobs and the OpenELEC fork LibreELEC are available as combinations. However, it turns out OpenELEC does not currently support an extension for activating the GPIO pins and has generally not been updated regularly since the fork three years ago. BerryBoot and LibreELEC do not always play well together. The developers therefore recommend avoiding this combination [10].

Several program add-ons are important for the use of the system. Advanced WoL is available for waking up the NAS, and Raspberry Pi tools are available for certain of its functions. The Audio Profiles add-on has also proven practical: It can be used to select which port to use for the output when a track or video is started. In this case, the HiFiBerry, the Bluetooth adapter, and the HDMI output are available.

Once the add-ons have been set up, you need to include the Python script in the LibreELEC autostart script [11]. You can also add some settings to the Raspberry Pi's config file [12] (e.g., the device tree overlay for the corresponding HiFiBerry version).

Because I am not supplying power to the Rasp Pi from the micro-USB input, but via the GPIO pins, a small flash constantly appears in the display. This

behavior normally warns of a power supply undervoltage, but not in this case. I thus decided to disable the display with the

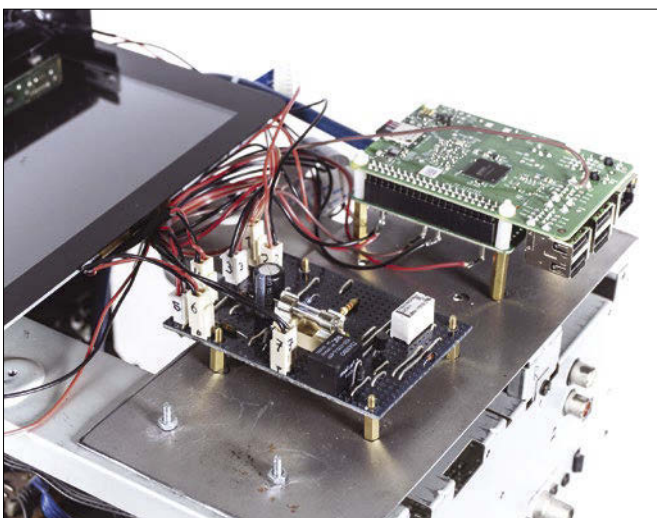
```
avoid_warnings=1
```

command [13]. In normal operation, the components, including the display, do not require more than 1A, so the 3A power supply I am using is fine (Figure 10).

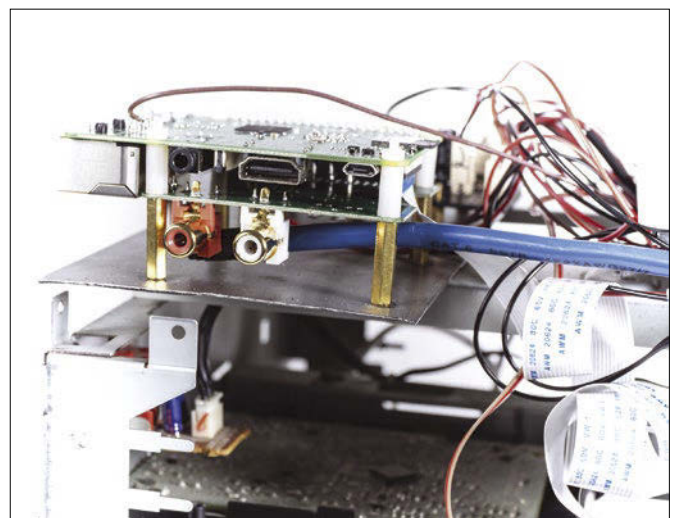
## Outsmarting Standby

The circuit still lacks one tiny thing in operation. The system automatically switches to standby mode if no key is pressed for 15 minutes at the end of a CD. Although the system starts up again after one or two seconds because it notices that the CD drawer (which no longer exists) is "open," the interruption disrupts operation. This prompted me to connect the cable formerly connected to the drawer button to another relay and use a Python script to simulate an occasional keystroke so the system no longer switches off.

The push-button (UNCAL) switch is also used: The graphics driver of the Raspberry Pi is not capable of powering two video outputs simultaneously (Figure 11). To output the video signal over HDMI to the TV, the power supply of the touch display has to be disconnected before boot, because the system does not support changing the display during operation. Finally, I installed a small LED on the front panel to show the system status.

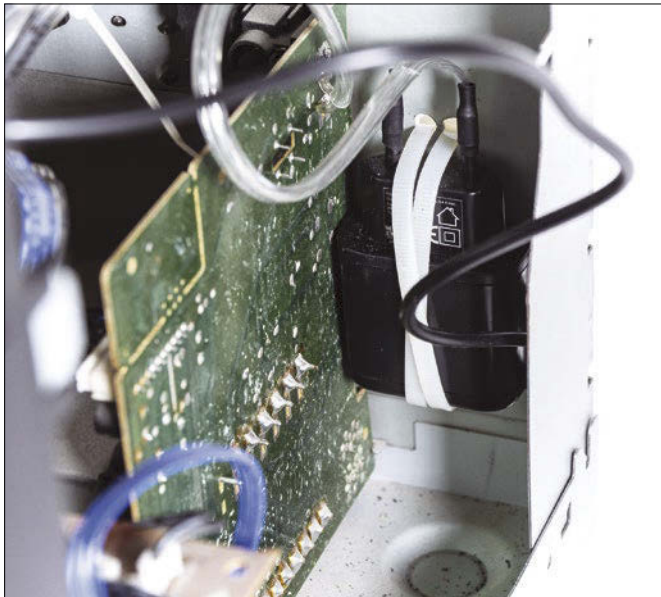


**Figure 8:** The Rasp Pi and its peripherals can be switched on and off safely with an additional circuit board.



**Figure 9:** The Rasp Pi and HiFiBerry boards fit into the corner of an unused CD housing with a metal plate underneath.





**Figure 10:** The power supply is 3A; it shares its connection with the internal power supply of the stereo system.

## Conclusions

In the end, the old diesel-powered VW didn't turn into a Tesla Model S, but the design met all of my requirements for a networked receiver – even waking up the NAS. The seven-inch touchscreen display [14] is not particularly large with its 800x480-pixel resolution, but it's still perfectly adequate for selecting music files and radio stations. Scrolling through long file lists takes some practice to hit the desired line, but frequently used folders and channels can be stored as favorites.

To view videos on TV, you need an app or a remote control. For my purposes, the

vantage of the removable display is that the microSD card can be exchanged quite easily on the Raspberry Pi and replaced with another installation without opening the housing.

The costs for the hardware amounted to about EUR190 (just over \$200). The touch display hit my wallet hardest at EUR70 (~\$75). The

free Kore [15] app, which connected to the Kodi installation over the network, was absolutely fine.

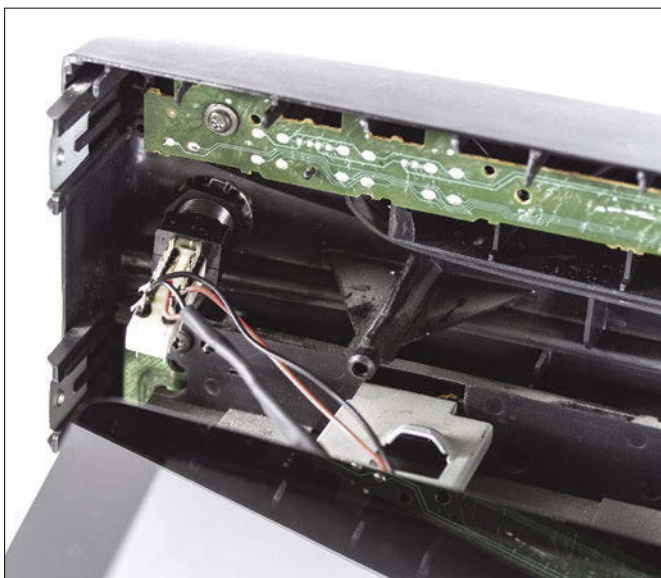
Compared with the tested receivers, my homemade version is not directly available at the push of a button. Booting with Noobs takes about 40 seconds, and a pure LibreELEC still takes about 20 seconds. The dual-boot system is not totally fit for the purpose in the end. One ad-

HiFiBerry DAC + Pro with two ultra-low jitter clock generators cost almost EUR40 (~\$45), including shipping, directly from the manufacturer. However, importing from Switzerland incurred additional costs. The Raspberry Pi 3 currently costs slightly north of EUR30 (~\$37), and the power supply about EUR10 (~\$11). Additionally, you need small parts for the circuit.

Granted, the tinkering involved here cannot be finished in one or two evenings, but to reduce the overhead, you could do without the on/off switch; then, the Raspberry Pi would boot as soon as grid voltage was applied to the system, and it could only be switched off from Kodi. However, I will not be disconnecting the two boards and the display from the power supply. The relay for outsmarting standby is also not mandatory if you connect the output of the sound card to the tuner or the AUX input, for example. When outputting by Bluetooth and HDMI, it does not matter anyway. ■■■

## Info

- [1] Compact system Philips FW 362: <https://www.manualslib.com/manual/776396/Philips-Fw-362.html>
- [2] Onkyo R-N855 instruction manual: [https://www.eu.onkyo.com/downloads/3/0/2/0/2/Manual\\_R-N855\\_En.pdf](https://www.eu.onkyo.com/downloads/3/0/2/0/2/Manual_R-N855_En.pdf)
- [3] FAZ test: [https://www.faz.net/aktuell/technik-motor/digital/netzwerkspieler-im-test-5-player-im-vergleich-15543747.html?printPageArticle=true#pageIndex\\_0&service=printPreview](https://www.faz.net/aktuell/technik-motor/digital/netzwerkspieler-im-test-5-player-im-vergleich-15543747.html?printPageArticle=true#pageIndex_0&service=printPreview) (in German)
- [4] HiFiBerry: <https://www.hifiberry.com>
- [5] Grundig 18 C compact system: <https://www.vintageshifi.com/repertoire-pdf/pdf/telecharge.php?pdf=Grundig-M-48-DC-Service-Manual.pdf>
- [6] HiFiBerry GPIO use: <https://www.hifiberry.com/build/documentation/gpio-usage-of-hifiberry-boards/>
- [7] Volumio: <https://volumio.org>
- [8] Euronorm standards: <https://en.wikipedia.org/wiki/Euronorm>
- [9] Samsung YouTube app: <https://www.myce.com/news/samsung-removes-youtube-app-smart-tv-models-2012-82000/>
- [10] BerryBoot and LibreELEC: <https://forum.libreelec.tv/thread/12896-libreelec-using-hifiberry-with-berryboot/>
- [11] Autostart script: <https://wiki.libreelec.tv/autostart.sh>
- [12] Rasp Pi config.txt: <https://wiki.libreelec.tv/config.txt>
- [13] Deactivate display: <https://www.raspberrypi.org/forums/viewtopic.php?t=177477>
- [14] Seven-inch touch display: <https://www.raspberrypi.org/products/raspberry-pi-touch-display/>
- [15] Kore: <https://kodi.wiki/view/Kore>
- [16] Kurt Tucholsky: [https://en.wikipedia.org/wiki/Kurt\\_Tucholsky](https://en.wikipedia.org/wiki/Kurt_Tucholsky)



**Figure 11:** A switch for the display lets you use the HDMI output for a TV.



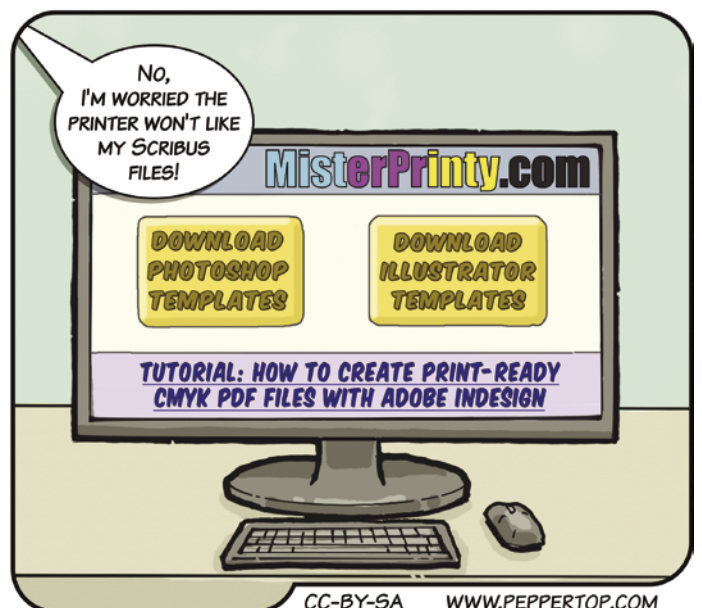
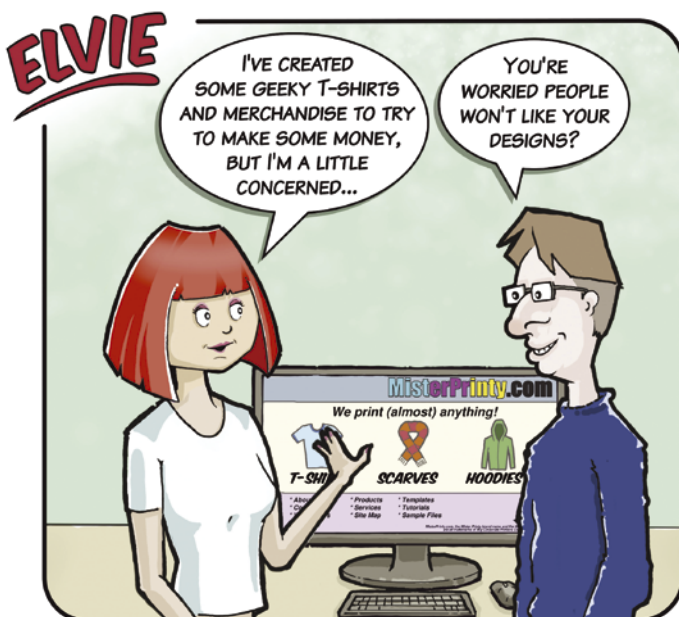
**Gaming on Linux** is a mixed bag. Many users adore the native open source games, and a thriving retro-gaming community carries the torch for low-tech gems of the distant past. But for many hard-core gamers, Linux is just a little more trouble. First of all, many game developers don't bother with putting out a Linux version. Tools like Wine provide a measure of compatibility, but configurations can be tricky and often require extra effort. The Lutris open gaming platform smooths out the kinks for gaming on Linux. Lutris offers automatic access to free and open source games and also provides an interface for accessing games from GOG, Steam, Battle.net, Origin, Uplay, and other gaming sources. In this month's LinuxVoice, we introduce you to Lutris. We'll also show you around McFly, an innovative utility that brings intelligent search to your Bash history, and we'll introduce you to some leading clients for the Mastodon free microblogging platform.



Image © Olexandr Moroz, 123RF.com

# LINUXVOICE ▶

<b>Doghouse – Licensing</b>	<b>73</b>
<i>Jon “maddog” Hall</i>	
For entrepreneurs with little money, FOSSH offers a way to get their projects off the ground.	
<b>Lutris</b>	<b>74</b>
<i>Tim Schürmann</i>	
If you frequently play games on Linux, you are accustomed to dealing with many different installers and configurations. Lutris can help simplify the process of setting up all your games.	
<b>McFly</b>	<b>80</b>
<i>Karsten Günther</i>	
McFly extends the Bash history's features and helps you find past commands more quickly.	
<b>FOSSPicks</b>	<b>84</b>
<i>Graham Morrison</i>	
As you might guess from certain titles in this month's selection, Graham has finally built himself an open source 3D printer.	
<b>Tutorial – Mastodon</b>	<b>92</b>
<i>Paul Brown</i>	
The open and simple Mastodon API makes it easy to create applications to interact with this federated microblogging platform.	



CC-BY-SA WWW.PEPPERTOP.COM

Shop the Shop

shop.linuxnewmedia.com

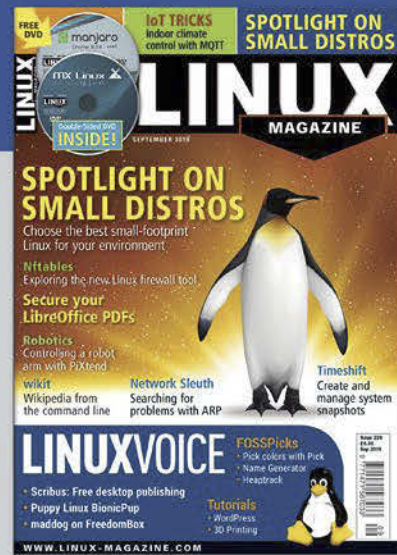
Want to subscribe?

Searching for that back issue you really wish you'd picked up at the newsstand?

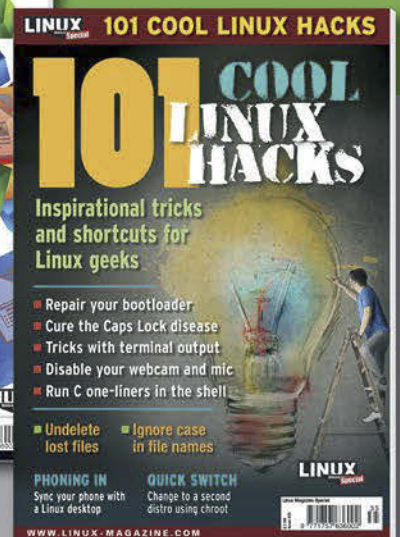
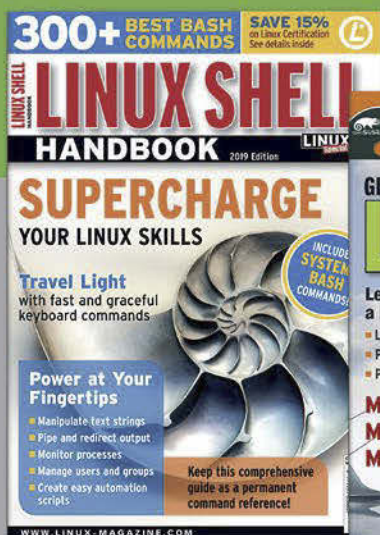
Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

shop.linuxnewmedia.com

## DIGITAL & PRINT SUBSCRIPTIONS



## SPECIAL EDITIONS





# MADDOG'S DOGHOUSE

For entrepreneurs with little money, FOSSH offers a way to get their projects off the ground. BY JON "MADDOG" HALL



Jon "maddog" Hall is an author, educator, computer scientist, and free software pioneer who has been a passionate advocate for Linux since 1994 when he first met Linus Torvalds and facilitated the port of Linux to a 64-bit system. He serves as president of Linux International®.

## Starting out with FOSSH

One of the nice things about my work with Free and Open Source Software and Hardware (FOSSH) is when people tell me how FOSSH has changed their lives and that I was the catalyst for that change.

It usually starts out with "I listened to you X years ago and now I am the CTO of my company" or "I listened to you Y years ago and now I am the CEO of a company and my company employs Z people," and their successes were tied to FOSSH.

Typically, when a person begins their career, they do not have a lot of money to start their company. The last thing they really need is to purchase a lot of expensive software licenses (and the lawyers needed to review the licenses and contracts) to allow them to build their first prototypes. FOSSH allows them to build those prototypes (and perhaps the actual finished product) without having to pay money upfront for those software licenses.

I was reminded of this at a conference in Kenya where a speaker told us that he and his other two colleagues only had \$30 to start their consulting business. By using free software, they could produce solutions for their customers without having to buy software licenses. However, they realized that as a new company they had no reference projects to convince potential customers to hire them. So they had to do three projects pro bono just to create examples of their work. After these three successful pro bono projects, they had the references to allow their company to get paying

jobs. Now, they are a successful multinational company.

Another speaker at the Kenyan conference discussed the cost of software licensing of commercial software versus FOSSH. While I knew that software licenses can be very expensive, the amount of money that he was talking about was absolutely breathtaking, often multiple millions of dollars. Even when he included service fees for FOSSH products, such as training and support fees, the amount of money saved from not buying the commercial closed source software was astonishing. Plus most of the licensing money went out of their country and did nothing for the local economy.

Another person told me that he had been trying to start his own accounting firm for five years, but he simply could not afford the commercial accounting software necessary for his business. Two hours after I had given my keynote on FOSSH, he told me that he had searched for FOSSH accounting software and found a package that gave him all he needed to start his business. He, of course, was very happy.

Unfortunately, I also have had some bad experiences.

One person told me that he had listened to me and that I had caused him to lose all of his money and go into bankruptcy. He explained that he had a great idea, borrowed some money, hired some engineers, created his product, then licensed it as FOSS, and no one paid him for it – they just used the code.

I told him that he had not listened. I always start my talks by telling people that they have to have a business plan of how they are going to make money with FOSS. He had not done that. Therefore, when he "gave away" his product, he had no plan to create sustainable income to cover his costs and make some profit. This was not my fault.

In discussions about license fees, I often talk about how much software piracy happens in various countries. In two countries where I recently had speaking engagements, Brazil and Kenya, desktop software piracy is reported at 84 and 76 percent respectively. A lot of the people in these countries think that software piracy is okay – until I ask them whether they would like their software that they wrote to be pirated. Then they get quiet. One young person approached me after my talk and told me he had never thought of his software piracy that way, and he was ashamed.

Despite the amount of piracy in these countries, they still send a huge amount of money outside of their country to pay software license fees, particularly from education, government, and large businesses.

As I write this, I am listening to a talk on cybersecurity where the speaker is lamenting that some of the cybercriminals take financial resources and send them "outside the country." In my mind, the same thing happens with commercial, closed source software license fees. It is legal, but the money is lost to the local economy in both cases. ■■■



# Install and manage games with Lutris

## Play Time

If you frequently play games on Linux, you are accustomed to dealing with many different installers and configurations. Lutris can help simplify the process of setting up all your games. **BY TIM SCHÜRMAN**

**G**ames and Linux are normally not a good match. Often a troublesome configuration or Wine issues make the setup a slow and time-consuming process. With Lutris [1], an open gaming platform, you just need a few mouse clicks to set up a new game on Linux.

When it comes to commercial software, Lutris will only install games that you have purchased in advance. However, Lutris usually automatically accesses free and open source games from online stores, such as Steam [2] and GOG.com [3]. All the games you install are stored in an integrated library; from there, the games can be started (or easily removed) at the push of a button.

### Helpers

Lutris uses Wine to launch numerous Windows games on Linux and takes care of the required Wine configuration independently. The program also cooperates with many other emulators, which you can use to launch countless classics, from the original Space Invaders to the Colonization strategy game for DOS.

Lutris uses runners, which are programs that run the games. For example, the Overwatch action

game launches with the Wine runner, while Colonization, the legacy strategy game, is driven by the DOSBox runner with a DOS emulator of the same name (Figure 1).

Native Linux programs have their own runner – unsurprisingly named Linux. The Lutris website [4] lists the currently supported runners, which include not only Wine and DOSBox, but also FS-UAE for Amiga and PCSX2 for Playstation 2 emulators, plus many more.

Lutris does not actually install games itself. Instead, it uses community-maintained installation scripts. A list of all currently supported games can be found on the Lutris website under the *Games* tab (Figure 2).

### Installation

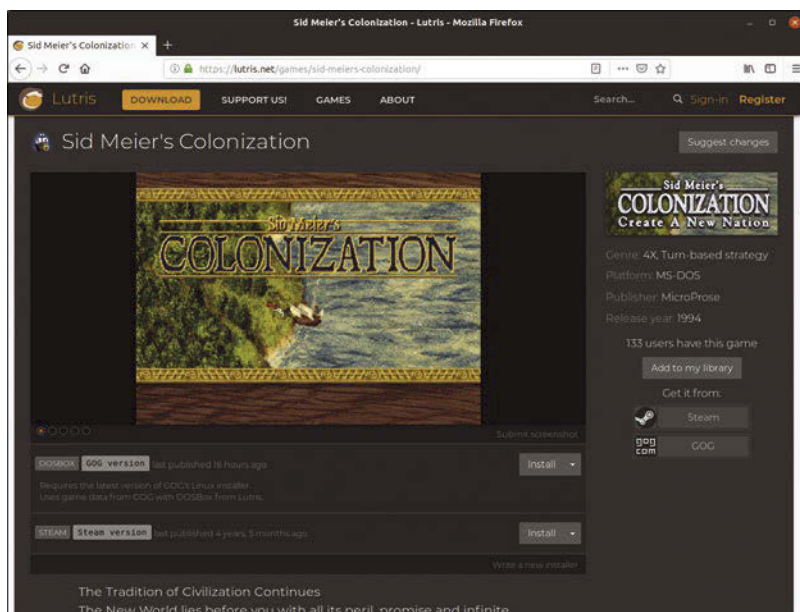
Some distributions, including openSUSE v15.1 and later, offer Lutris via their software package management. However, there are plenty of community packages available for older releases. The Lutris developers provide their own Ubuntu Personal Package Archives (PPA) repository. Listing 1 shows the three commands that add the repository and install the current Lutris version.

If the distribution you are using does not provide a Lutris package, download the `tar.gz` archive from the *Download* section of the Lutris website and extract it to disk; installing is not required.

Before you install Lutris, you need to make sure that you have Wine and all other required dependencies installed on your system (see the “Requirements” box). You can find these dependencies via the software package administration. On Ubuntu, the command from Listing 2 installs all necessary dependencies. You can then launch Lutris by calling `./bin/lutris` from the program’s directory.

Some games use the Vulkan graphics interface. If Lutris complains about missing Vulkan libraries at startup, you need to install these libraries man-

**Figure 1:** Clicking on a game in the *Games* tab on the Lutris website provides you with details about the game, including known issues. You can also purchase the game directly from GOG or Steam by clicking on the corresponding link.



#### Listing 1: Installing Lutris on Ubuntu

```
$ sudo add-apt-repository ppa:lutris-team/lutris
$ sudo apt-get update
$ sudo apt-get install lutris
```

## Requirements

Lutris requires a number of tools and libraries. To run the Lutris client, you need to install:

- Python 3.4 or higher
- *PyGObject*
- *PyGObject* bindings (GTK, GDK, Gnome Desktop, WebKit2, and Notify)
- *python3-requests*
- *python3-pillow*
- *python3-yaml*
- *python3-evdev* (optional for controller detection)

To install and run games with Lutris, you need the additional packages:

- *PSmisc* (or a package for *fuser*)
- *p7zip* (or a package for *7z*)
- *cURL*
- *fluid-soundfont-gs* (or other soundfonts to play MIDI music)
- *cabextract* (for installing Windows games)
- *xrandr* (for systems with an X11 server)
- *libc6-i386* and *lib32gcc1* (for 32-bit game support)
- A 32-bit OpenGL graphics card driver

ually. The procedure depends on which graphics card and distribution you use. (Describing this process is beyond this article's scope; for detailed instructions, see the Lutris Wiki [5]).

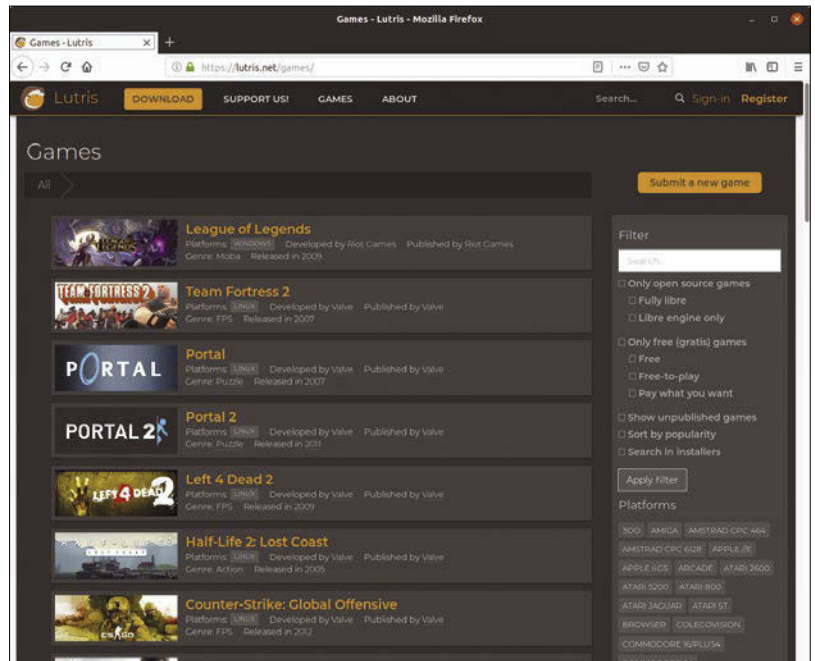
## Runners

In the initially empty main window, Lutris displays all the currently active runners in the left sidebar (similar to the sidebar in Figure 3). At first, only the *Browser* and *Linux* runners should appear. These allow Lutris to install native Linux programs and run browser games. To add more runners, click the gear icon to the right of *Runners* and select the desired runners from the list (Figure 4).

If you want Lutris to start and manage Windows games, you need the *Wine* runner (located at the bottom of the list). Click on the adjacent green installation button. With *Wine*, you can even select specific versions. Unless you have a reason, you should always select the highest version number without the *tkg* prefix.

Lutris stores the downloaded runners in the home directory below `.local/share/lutris/runners/`, smuggling them past the distribution's package manager. Install all the other required runners in the same manner. The *DOSBox* runner for DOS classic games is especially useful.

If you want to play games from Steam, then you also need the *Steam* and *Wine Steam* runners. *Steam* is responsible for native Linux games on

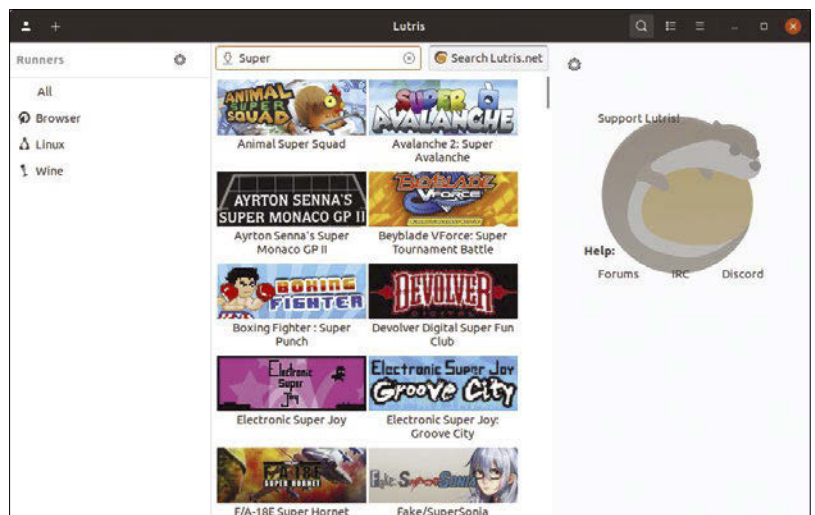


**Figure 2:** Lutris only installs games that it supports; you can find a current list on the Lutris website.

Steam, while *Wine Steam* launches Windows games offered on Steam. When setting up *Wine Steam*, your system may complain about missing *Wine-Mono* and *Wine-Gecko* packages; install them if necessary.

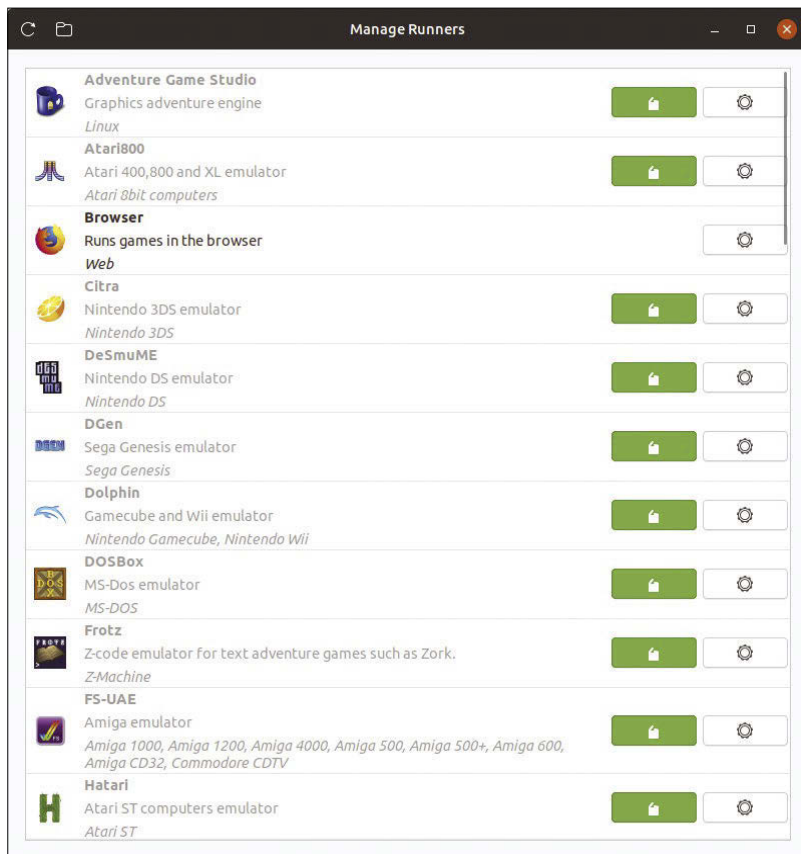
Incidentally, the runners are the Steam client for Linux or Windows, which Lutris uses for its own purposes. On our test system, Lutris did not want to install the *Steam* runner. In this case, you need to manually install the Steam client for Linux, and Lutris will automatically detect it the next time it starts.

**Figure 3:** Runners for Linux, Windows, and browser games are available here. In the center pane, Lutris lists all the known games that include the *Super* string in their title.



## Listing 2: Installing Dependencies on Ubuntu

```
$ sudo apt install python3-yaml python3-requests python3-pil python3-gi
girl.2-gtk-3.0 girl.2-gnomedesktop-3.0 girl.2-webkit2-4.0
girl.2-notify-0.7 psmisc cabextract unzip p7zip curl fluid-soundfont-gs
x11-xserver-utils python3-evdev libc6-i386 lib32gcc1
libgirepository1.0-dev
```



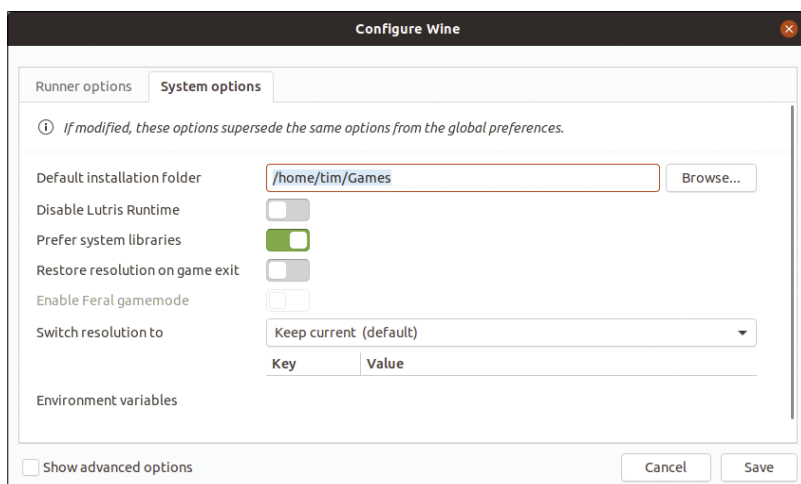
**Figure 4:** Lutris mainly has runners for emulators.

To delete a runner, click on the red trash icon next to the runner in the *Manage Runners* window.

### Changing Paths

In the main window, you will find a gear symbol at the top of the sidebar on the right (above *Support Lutris!*). Clicking on it opens the basic settings. You don't have to change anything, unless you want to change the default installation directory. When setting up a game, Lutris automatically suggests an installation directory based on the game's name and locates it under the `~/Games/`

**Figure 5:** In the settings for a game or a runner, you can see additional options by checking the *Show advanced options box* (bottom left corner). These options are helpful if a game fails to start.



folder by default. To change this directory, in the *System options* tab, specify your own *Default installation folder*, overwriting the runner's default (Figure 5). In the example shown in Figure 5, games powered by Wine would all end up in a separate folder. This is only recommended if you are installing a large number of games.

In the left sidebar of Lutris's main window, a gear icon again appears when you mouse over a runner entry. Clicking on the gear icon reveals the settings for the respective runner. In the case of Wine, you have the option of changing the *Wine version*.

### Installing Games

To install a game, click on the magnifying glass icon, then on the *Search Lutris.net* button, and type the name of the game in the input field. Lutris searches its games database and displays all matches in the middle pane (Figure 3).

Even with a fast Internet connection, the search can sometimes take a few seconds. Once you have found the game you desire, click on the corresponding icon image. Pressing *Install* in the right column sets up the game.

Sometimes Lutris will offer several versions of a game, such as the open source SuperTuxKart game (Figure 6). Click the *Install* button next to your chosen version. For commercial games, Lutris offers you the choice of downloading the game from various online stores.

After selecting the desired game, follow the instructions on the screen. The game may require additional software or a specific runner, which you can launch by clicking *Install*. If there are several runners to choose from, the first matching suggestion is usually found at the top of the list.

If you do not want to change this, simply accept the suggested installation directory by pressing *Install*. A separate window shows the progress of the game setup. If this bothers you, just drag the window to another place on the desktop, and you can continue working with Lutris. When the installation is complete, *Launch Game* starts the game.

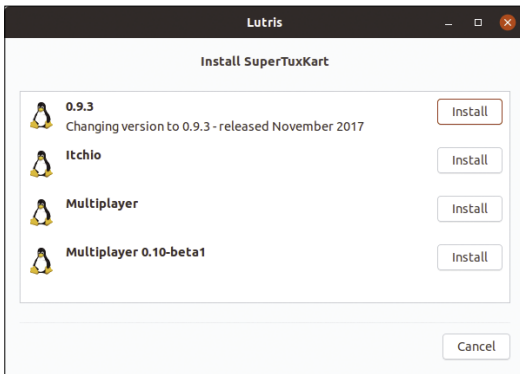
### Librarian

If you hide the search function by clicking on the magnifying glass icon, Lutris will show you all the currently installed games. A click on one of the runners limits the display to all games belonging to that runner. If you use games for different platforms, such as Windows or DOS, select the appropriate option bottom left.

If you select *All* in the Runners sidebar, Lutris will display all the games again. To search locally for a specific game, open the search function with the magnifying glass, click the *Search Lutris.net* button again, and enter the name of the game.

The icon to the right of the magnifying glass switches to the list display and back again if re-





**Figure 6:** The SuperTuxKart *Itchio* version is downloaded from itch.io instead of the Lutris website.

quired. You can use the hamburger icon to change the software's display options (Figure 7). The slider at the top of the hamburger pop-up menu changes the size of the icons, and the checkboxes in the middle part of the menu specify the sort order. In the default setting, the program lists all games alphabetically in ascending order by name.

As soon as you click on a game, further actions appear on the right side. *Play* starts the game; clicking on the trash can uninstalls it. Using the appropriate links, you can create an icon on the desktop and an entry in the start menu if required. A click on the folder icon opens the game's installation directory in a file manager, for example, to let you save high scores.

The gear symbol takes you to various settings. In the *Game options* tab, you can enter arguments for the game in the *Arguments* field. Click on *Game info* to exchange the button label image.

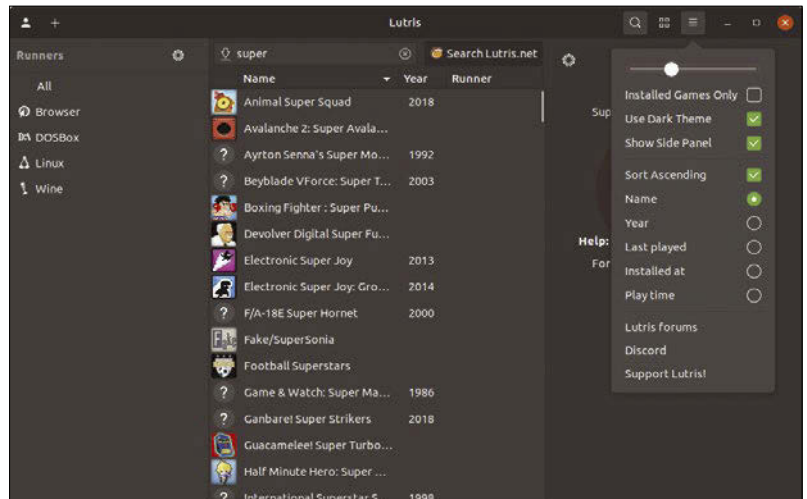
If the game fails to launch, switch to *System options* and enable *Disable Lutris Runtime*, which means that the software won't load various libraries before the game starts. If this is not enough, disable *Prefer system libraries*. The game then uses its own libraries instead of those belonging to the system.

## Adding Games

If you want to add manually installed games to your collection, click on the plus icon in the upper left corner, select *Import games*, and make sure that you are in the *Desktop games* tab with the penguin icon. Lutris will show you all the games it found here (Figure 8). To introduce Lutris to one of the games, check the box in front of the game name, then press *Import games*, and close the window.

If you have previously installed games via the Steam client, switch to one of the tabs with the Steam icon instead. Select all the games installed with the Steam client that you want to manage with Lutris, and then click *Import games*.

To import games purchased from GOG, go to the GOG.com tab and click *Connect your account*. Log in to GOG.com as usual. It then takes a moment for



**Figure 7:** A list display and a dark theme (enabled by checking the *Use Dark Theme* box) were chosen here.

Lutris to read your library. Then follow the steps described for the Steam games above. In the main window, you will now find all the games that you set up in the usual way using *Install*.

## Sharing

If you play on multiple computers, you can synchronize your game list between Lutris installations via a free account at Lutris.net. Click on the *Register* button located top right on the website, and then follow the instructions on the screen. After doing this, log in via *Sign-in*.

Go to the *Games* tab, and click on a game to add it to your list (*Add to my library*). You can view the list by clicking on the Lutris logo top right and selecting *Show library*. In the main Lutris window, click on the Lutris logo top left and log in with *Login*. Now use the Lutris icon to synchronize the current status via *Synchronize Library*.

## Conclusions

We tested Lutris v0.5.2, which was a little shaky on our Ubuntu system from time to time. Games did not always start at the first attempt. Nevertheless, Lutris helps you to get an overview of your game collection and also considerably simplifies the process of setting up games. ■■■

**Figure 8:** The games listed in the *Desktop games* window were probably installed via the system's software manager.



## Info

- [1] Lutris: <https://lutris.net>
- [2] Steam: <https://store.steampowered.com/>
- [3] GOG.com: <https://www.gog.com/>
- [4] Runners: <https://lutris.net/runners>
- [5] Driver installation: <https://github.com/lutris/lutris/wiki/Installing-drivers>



**LINUX MAGAZINE**  
MAY 2013

MANAGING PDFs in LibreOffice  
SNAPSHOTS PROTECT YOUR DATA

**SNAPSHOTS**  
A better way to protect your data?  
Plasma 5.14  
Exploring KDE's new age desktop  
Optimize Gimp Images with Python Plugins  
tmate  
Emergency repairs from the command line

New in Bash  
Cool new commands for the terminal window

WIFI Airboat  
Plastic bottles, duct tape, 2 Internet connections

OpenWRT  
2 Internet connections on the router

FREE DVD  
kubuntu  
CentOS 6.4  
LINUX  
Double-Sided DVD INSIDE!

**LINUX MAGAZINE**  
JUNE 2013

FreeBSD 12.0  
Gnome Remote Desktop  
Remote sharing for Wayland-based systems

**GIT TRICKS**  
VERSION CONTROL SECRETS FROM THE EXPERTS

Choose a Git GUI  
Bash 5.0  
Long-awaited update for the classic hacker shell  
FreeBSD 12.0  
Will this excellent server OS work as a desktop system?  
VolkPC  
Run a Debian desktop over Android  
Google Firebase  
Cloud storage for your IoT project  
Sudo Voodoo  
Fine-tune system privileges

**LINUX MAGAZINE**  
JULY 2013

PROJECT FOSS phone for the masses  
ENERGY CONSUMPTION  
WHY ARE SOME APPLICATIONS MORE EFFICIENT?

**ENERGY CONSUMPTION**  
Why are some applications more efficient?  
Network Analysis  
Look for intruders with Wireshark and Netflow  
File Encryption with EncFS  
Picture Perfect  
We compare 5 image viewers  
Maker Tricks  
Analyze malware on a hacked Rasp Pi  
mitproxy  
Troubleshooting HTTPS connections

FREE DVD  
fedora  
ubuntu  
LINUX  
Double-Sided DVD INSIDE!

**LINUX MAGAZINE**  
AUGUST 2013

Light monitoring with a Raspberry Pi  
Hack on the cheap  
Hack your smart plugs and make them talk to Linux

**IoT on the Cheap**  
Hack your smart plugs and make them talk to Linux

Find Performance bottlenecks with eBPF  
Karoshi  
The easy way to set up a complex server system

Timer Tools  
Schedule commands and scripts with at, cron, and anacron  
chorny  
Sync up with the other time service daemon  
Go Tricks  
Search for photos by GPS coordinates

FOSSPicks  
Kdenlive  
Kodi  
8...  
Tutorials  
KDE Partition Manager  
Olivia Cloud Music Player  
DOSBox X  
Singing: Add User Input  
Preparing an Object for Printing

Count Your Money with GnuCash  
Favorite Time Management Tool

**LINUX MAGAZINE**  
OCTOBER 2013

HUGE SAVINGS! 12% OFF VALUE!  
DVD INSIDE! ALL 214 ISSUES ON A SEARCHABLE DISC!

**PRIVACY**  
STOP SNOOPERS AND PROTECT YOUR IDENTITY

spoozers and protect identity  
your music player with free firmware  
organize all your news sources  
aces for your Rasp Pi  
Ready to Go?  
Smart queries in the powerful Go language

FOSSPicks  
Surge Synth  
Kloak Keystroke anonymizer  
Perspolis

**LINUX MAGAZINE**  
NOVEMBER 2013

Up close with the classic Linux version control system  
GIT WORKSHOP  
The benefits of distros on Linux  
.NET CORE

**.NET CORE**  
Microsoft's flagship work on Linux

Little-known OS to compete with BSD?  
and-mapping Tools  
Visualize decisions and organize your best ideas  
Counting Pennies  
Tabulating audience feedback with a Rasp Pi  
Cool new...  
Remote Pair Programs  
ExaGeat  
Run x86 apps on Rasp Pi and other ARM systems

Electron Framework  
Cross-platform apps with JavaScript and HTML  
Remote Pair Programs

**LINUX MAGAZINE**  
DECEMBER 2013

Ultra-safe Linux with routine protection  
QUBES OS  
INNOVATIVE DISTROS

Are these hidden gems:  
- A simpler enlightenment  
- Anonymous surfing even on hardware  
- All designers get creative with KDE  
- OS - Lockdown security in the focus on isolation  
cheatsh  
Syntax tips at your fingertips  
Filesystem  
down into a mining system  
Upribox 2  
Use a Rasp Pi to filter ads and trackers  
Git Workshop  
Working with remote repositories  
Purism Librem  
Linux Laptop special security features

**LINUX MAGAZINE**  
JANUARY 2014

TIPS FOR USING CASCADING STYLE SHEETS  
NAVIGATION TOOLS FOR EXTENDING OPENSTREETMAP

**NAVIGATION**  
Free services that extend OpenStreetMap  
Kernel Disposal  
Are old kernels sucking up valuable space on your hard drive?  
Build a Robot Car  
Zorin OS  
Will this well-appointed Linux succeed in attracting Windows users?  
WebAuthN  
Authenticate without a password  
Put It On!  
Build your own wearable with Android and MIT's App Inventor  
DOOM Killer  
What happens when kernel runs out

FOSSPicks  
whiskerhd  
musicube  
Imaginary Teleport

Tutorial  
Getting started with KDE's Plasma

**LINUX MAGAZINE**  
FEBRUARY 2014

Ubuntu 18.10  
Downloads 3.10.1  
Double-Sided DVD INSIDE!

**CUSTOMIZE THE BOOT MENU**  
Clean up your startup for clarity and fewer mistakes  
iCloud SSDs  
Make the storage fit your hardware  
IoT Tricks  
Store data in memory with Redis  
Exploring Ubuntu 18.10  
Pulse Sensor  
Measure your heartbeat with a Raspberry Pi  
EncryptPad  
Text editor with easy encryption

FOSSPicks  
Sweet Home 3D  
Tentaculus  
Flare Emptees Campaign

Tutorials  
Shell Variables  
Natron Video Effects

**LINUX MAGAZINE**  
MARCH 2014

Save time and extend the power of your virtual machines  
Scapy  
Automate packet analysis with Python  
Put a Recording Studio on a Raspberry Pi  
Elementary OS  
Elegant Linux with an ambitious vision  
Create a Cartoon  
With open source tools  
Tools for Writers  
Find words and organize your thoughts  
PDF Tricks  
Clean up blotches and fading text

VirtualBox Hacks

FOSSPicks  
eDEX-UI  
Aurigo  
Calculator

Tutorials  
Bash Arrays  
Natron Video Tricks

**LINUX VOICE**  
FOSSPicks  
Visual Podcast Manager  
MUSC - Like-Sound BBS  
Leo Online Editor  
Tutorials  
RSS feeds

Astronomy with OpenAstr  
maddog - BSA and the real cost of Linux Software

**LINUX VOICE**  
FOSSPicks  
Retrosahre - Secure communication over the Internet  
maddog - IoT in Brazil  
Military-Malware Complex - Security: The search for zero days

Tutorial  
Getting started with KDE's Plasma

**LINUX VOICE**  
FOSSPicks  
EasySSH  
Saving Analog Data  
maddog: RISC-V Architecture

Tutorials  
Shell Variables  
Natron Video Effects

**LINUX VOICE**  
FOSSPicks  
Go For It! Easy Tool for building ToDo lists  
maddog: Diligence and other FOSS resolutions for 2019

Tutorials  
Bash Arrays  
Natron Video Tricks



**Linux Magazine** is your guide to the world of Linux. Look inside for advanced technical information you won't find anywhere else!

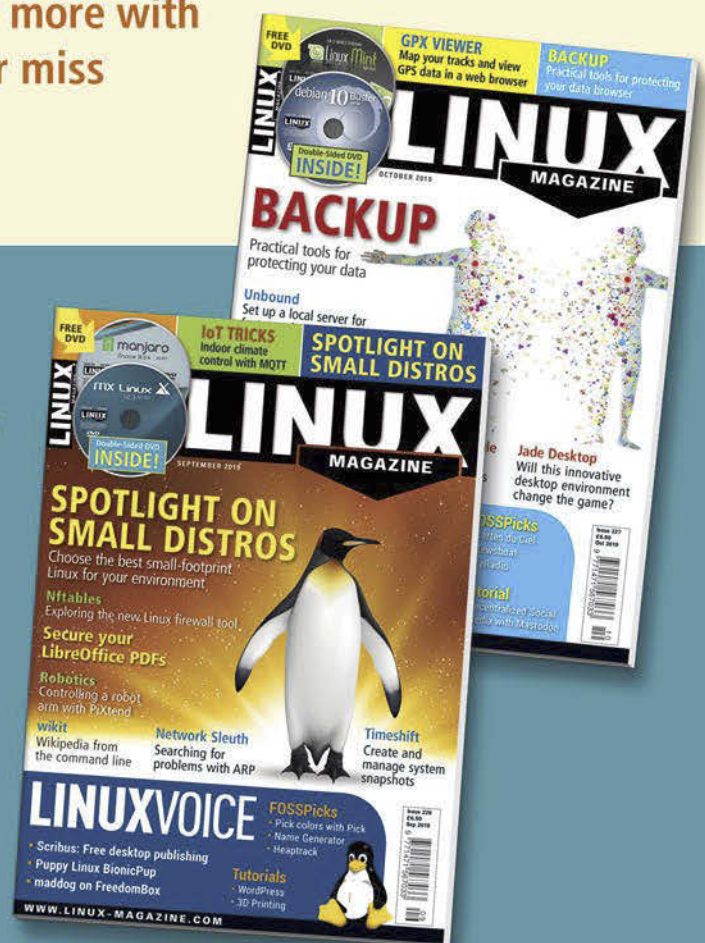
### Expand your Linux skills with:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

If you want to go farther and do more with Linux, subscribe today and never miss another issue!

## Subscribe now!

[shop.linuxnewmedia.com/subs](http://shop.linuxnewmedia.com/subs)





# McFly upgrades Bash with artificial intelligence

# Intelligently Arranged

When it comes to working at the command line, using Bash history effectively can save you time. McFly extends the Bash history's features and helps you find past commands more quickly. **BY KARSTEN GÜNTHER**

The most popular Linux shell by far is the GNU Bourne-Again Shell (Bash for short). One of its many outstanding features is its integrated command history, where Bash saves previously run command lines so that you can recall them later on without a lot of typing.

Listing 1 shows the output you'll see if you enter the `history` command at the terminal prompt. Each line begins with a line number in ascending order. The last command line to have been entered is at the end of the history list. You can use the arrow keys to scroll backward or forward one step in the history in the terminal. Pressing the Enter key copies the displayed line and runs it again.

Bash supports two approaches to searching for specific strings directly in the history: `Ctrl+R` searches backwards from the current cursor position; `Ctrl+S` searches forwards from the current cursor position. `Ctrl+S` only makes sense if you have already advanced further back in the history and are no longer at the end. In addition, entering `Ctrl+S` only works in an appropriately configured terminal; normally, this keyboard shortcut blocks the terminal.

To indicate reverse searching in the history, Bash uses `Ctrl+R` with a special prompt:

```
(reverse-i-search)`':
```

`i-search` stands for an incremental search, which refines or enhances the search by entering a new character. The function finds the last line in the history that matches the search criteria; each time `Ctrl+R` is pressed, the search jumps one match further back into the past.

In practice, this form of search in the history proves to be as simple as it is effective. A few cleverly selected entries are usually sufficient to find the desired line. However, you need to know what you are looking for – and, above all, you need to know a good pattern that will quickly take you to the desired command. The pattern can be anywhere on the command line, so you don't have to type it from the beginning. (See also the "Search Patterns" box.)

## Search Patterns

You can combine `history` with `grep` to find all command lines in the history where a given search pattern occurs (Listing 1, line 17). Using `agrep -B` instead of `grep` gives you a fuzzy search.

## Listing 1: Running the history Command

```
01 $ history
02 [...]
03 446 nano .config/GIMP/2.10/scripts/image-subdivide.scm
04 447 nano .config/GIMP/2.10/scripts/slice-and-join.scm
05 [...]
06 901 yay -Syu
07 902 trizen -Syu
08 903 yay -Syu
09 [...]
10 1044 ++
11 1045 ++ /tmp/
12 1046 mv /tmp/*png . -v
13 1047 ++
14 1048 mv /tmp/*png . -v
15 1049 ++
16 1050 mv /tmp/linify.png .
17 $ history | grep slice
18 447 nano .config/GIMP/2.10/scripts/slice-and-join.scm
19 1051 history | grep slice
```

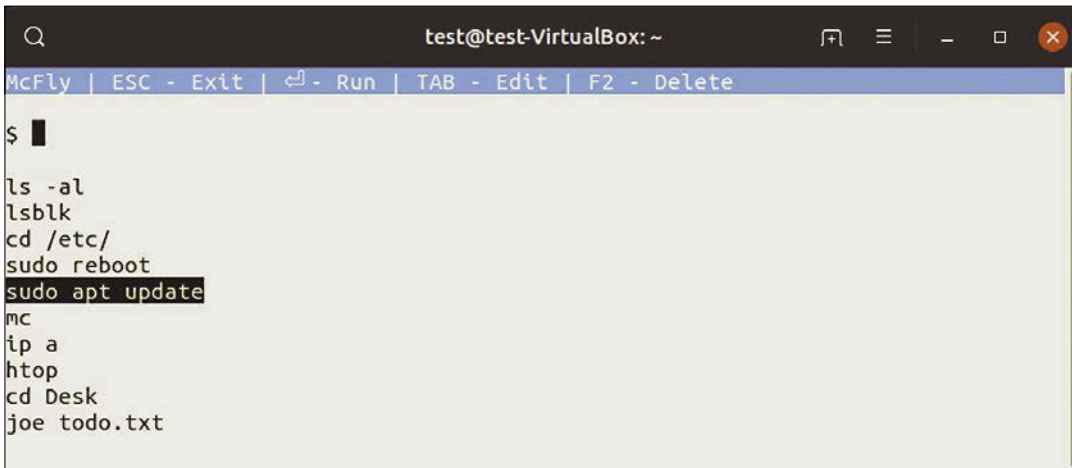


Figure 1: McFly extends the search for commands in the shell's command history.

For instance, to retrieve the command from Listing 1 (line 16)

```
mv /tmp/linify.png .
```

you would just need to type `Ctrl+R /FY`. Command lines such as `trizen -Syu` or `yay -Syu` can be quickly found with `Ctrl+R, Shift+S`, or `Ctrl+R - Shift+S`.

As shown in Listing 1, the history usually contains command lines that are very similar to each other and can be described well with patterns, but cannot be easily distinguished. A remedy for this requires a new approach.

### Modern Times

Bash history lacks the ability to search in a context-based way. For example, on many modern

systems based on `systemd`, `/var/log/journal/` contains large numbers of logfiles. Deleting outdated logfiles can often free up a large amount of storage space.

To see if, and how many, old logfiles are stored there, change to the directory. The `du -sh` command then determines the space used by the existing files. `find -mtime 7` lets you search for all files that have not been changed in over one week; you can then delete them using `rm [...]`.

You could now write a script or – preferably – a shell function for this task in a few moments, but the problem remains the same: You have to actively develop a solution for every situation. Wouldn't you prefer that the shell guessed what you were going to do based on the directory and offered you a corresponding action?

### Installing McFly

McFly [1] is not yet found in popular distribution's package sources. Only Arch Linux offers the program in the Arch User Repository (AUR). On Arch, you can install the program with an AUR helper, for example, using

```
yay -S mcfly
```

This will add a whole series of additional packages with a total volume of several hundred megabytes due to the dependencies.

In Ubuntu, and many other distributions, you need to install the Brew program, which was actually developed as a package manager for Mac OS X and later ported to Linux. Depending on the distribution, the package in question goes by the name of *brew*, *linuxbrew*, *homebrew*, *homebrew-bundle*, or – like in Ubuntu – *linuxbrew-wrapper*. Using the commands from the first six lines of Listing 2, you import Brew in Ubuntu and configure the program. Now set up the `tap` with Brew (line 7), which retrieves several hundred mega-

bytes of additional software. Following this, install by typing the command on line 8, which you then have to initialize (line 9).

Finally, expand your `$PATH` to include the executable file's folder. To do this, transfer the code in Listing 3 to the `~/.bashrc` file. After a restart, you will be able to call the `mcfly` command from the terminal.

Everything went smoothly on my Ubuntu test system, but some things can go wrong with this procedure: The `/home/linuxbrew/` directory and its `.linuxbrew/` subfolder may not be created automatically. When running `brew install mcfly`, you will then see an error message. In this case, you need to create the directory yourself or create a symbolic link to it. (See also the "Bug Report" box.)

The McFly developer has also posted ready-to-go 32- and 64-bit binaries of the program on GitHub [2].

**Listing 2: Configuring and Installing Brew**

```

01 $ sudo apt install linuxbrew-wrapper
02 $ brew
03 $ echo 'PATH="/home/linuxbrew/.linuxbrew/bin:$PATH"' >> ~/.profile
04 $ echo 'MANPATH="/home/linuxbrew/.linuxbrew/share/man:$MANPATH"' >> ~/.profile
05 $ echo 'INFOPATH="/home/linuxbrew/.linuxbrew/share/info:$INFOPATH"' >> ~/.profile
06 $ PATH="/home/linuxbrew/.linuxbrew/bin:$PATH"
07 $ brew tap cantino/mcfly https://github.com/cantino/mcfly
08 $ brew install mcfly
09 $ source "$(brew --prefix)/opt/mcfly/mcfly.bash"

```

**Listing 3: Expanding \$PATH**

```

if [ -f $(brew --prefix)/opt/mcfly/mcfly.bash ]; then
. $(brew --prefix)/opt/mcfly/mcfly.bash
fi

```

McFly [1] does this by reading the previous history, analyzing it in a synthetic neural network, and attempting to recognize correlations in the command lines by machine learning. These relationships then serve as the basis for the suggestions offered by the shell's search functions.

In contrast to the neural network's quite complicated structure, McFly's last step is very easy to understand: The `previous-history` function normally mapped to Ctrl+R is replaced by the new `mcfly search` function.

The following parameters serve as the basis for the neuronal evaluation of the history and the McFly suggestions for the search function, in the specified order:

- The current directory
- The previously entered command lines
- The frequency and order of the call
- Whether a command line was previously found by McFly

■ Whether the command ran without errors  
After setting up McFly (see the box "Installing McFly"), the shell behavior changes. Ctrl+R calls the tool directly; the titlebar changes to reflect this, and the new key bindings become active (Figure 1).

While initializing, McFly reads the existing history and evaluates it, which takes some time. You can press Esc to exit this mode and return directly to the prompt. Then use the up and down arrows to navigate the recommendations offered by McFly. When you enter a search term, McFly retrieves matching hits from the history.

Press Enter to accept the currently highlighted line from the history and execute the command immediately. The Tab key, on the other hand, transfers the highlighted history line to the input prompt without executing the com-

mand immediately. You can then edit the command and press Enter to execute it. Press F2 to delete individual lines from the list of suggestions after a prompt.

**Artificial Intelligence**

The interesting thing about McFly is its underlying artificial intelligence, the neural network. What used to require a huge amount of power and entire data centers, can now (with some limitations) even be performed by a Raspberry Pi. Today, such small neural networks are useful for all kinds of tasks, from image processing to character and pattern recognition.

These systems' missing logic often proves to be problematic. Neural networks learn coherencies without it being possible to clearly understand

**Bug Report**

If McFly crashes on your system with a cryptic message similar to:

```

Unable to add cmd_tpl to commands: 2
  SqliteFailure([...])', 2
src/libcore/result.rs:1009:5

```

you are struggling with a bug that has affected many other users [3]. Commands with special characters will disrupt the program. In the test this happened when I accidentally entered the command:

```
^@^@[...]^@^@sudo
```

Until the developers fix this problem, you'll have to make do with simply deleting the line from `~/.bash_history` with an editor.



how they do it in each case. With McFly, however, you can still see the learning behavior, or at least the rudiments of it. If you call a command from the list of suggestions, McFly moves the command further up the list the next time it is called. However, it is less clear which commands appear (and in which order) initially in the proposal list. The rules used already seem so complex that some effort is needed to predict their impact.

You actually encounter a very similar system every day when searching on the Internet. Here, too, it is often impossible to identify how, where, and why certain results appear in the hit list while others do not. This problem is attributable to the principle of neural networks and is not completely harmless. For this reason, IT researchers repeatedly urge cautious use of the technology [4].

### Conclusions

In addition to autocompletion, the history is an important shell feature for interactive use. McFly adds value to the `history` command. Fortunately, the developer implemented the tool as an independent program instead of adding it to the already complex Bash or developing a plugin. This minimizes the side effects and allows the tool to be tried out quickly and safely.

As far as McFly's AI performance is concerned, a typical problem of self-learning systems applies here. It is virtually impossible to make reliable statements about the software, since using the software changes the system itself at run time. In my test, McFly delivered very good results; most of the command lines I searched for were found in the first list of suggestions.

If you are just looking for an upgrade of `history`, there are simpler alternatives in the open source universe, such as `hstr` [5] and `cdhist` [6], which do without neural networks and take up far less disk space. ■■■

### Info

- [1] McFly: <https://github.com/cantino/mcfly>
- [2] McFly binaries: <https://github.com/cantino/mcfly/releases>
- [3] SQLite3 error: <https://github.com/cantino/mcfly/issues/56>
- [4] AI potential harm: [https://en.wikipedia.org/wiki/Artificial\\_intelligence#Potential\\_harm](https://en.wikipedia.org/wiki/Artificial_intelligence#Potential_harm)
- [5] `hstr`: <https://github.com/dvorka/hstr>
- [6] `cdhist`: <https://github.com/bulletmark/cdhist>

# FIND THE RIGHT TOOL!

## Check out our new Linux Administration corner!

<http://www.linux-magazine.com/administration>

SPONSORED BY



Linux  
Professional  
Institute



# FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software



As you might guess from certain titles in this month's selection, Graham has finally built himself an open source 3D printer. **BY GRAHAM MORRISON**

## Space visualization

# Gaia Sky

Space is big. This is perhaps why we have a large choice of astronomy and planetarium applications. Both Stellarium and KStars are fantastic, for example, along with Cartes du Ciel, which we looked at last month. But space simulation and visualization is a slightly different and more niche genre, and one that isn't so well populated. The incredible Celestia lets you fly through an ultra-realistic 3D space environment at faster than the speed of light, but its development pace has slowed to a glacial

speed over the last few years. This is why it's wonderful to find Gaia Sky, an even more impressive portal into not just the night sky, but the entire universe. And it's being actively developed.

Gaia's capabilities quickly become apparent after launching the application. You first need to decide on which texture size to download, with high-res textures requiring an additional 250MB. This is nothing compared to the star catalog; the complete star catalog is a staggering 60GB, and you can aug-

ment this with various other catalogs and 3D meshes. If you know anything about space exploration, this catalog size may sound familiar, and that's because the application has been built and developed by the European Space Agency (ESA), bundling data from its own Gaia mission to chart the roughly one billion stars that surround us in our galaxy.

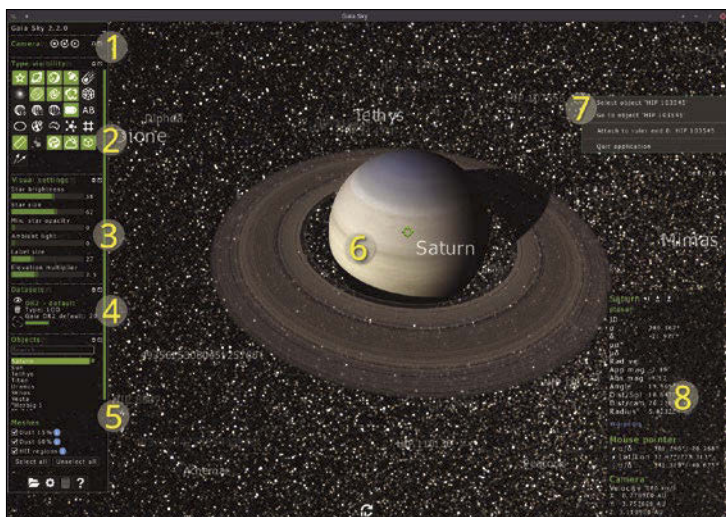
The main view starts with a celestial perspective on mother Earth. Much like many astronomy applications, you can use the mouse to change your perspective, and our planet is lit accurately to reflect the time of day on our sphere's light and dark sides. It all looks very tranquil from this distance. The backdrop is lit by the millions of stars found in your chosen catalog, and the brilliant thing about Gaia Sky is that you can not just click on these points of light to find out more information, but actually visit any of them. Right click and select *Go to*, and you immediately travel through space to your chosen location, from where you can view your alien surroundings. This being the ESA, star positions are accurate, but so too are their proper motions and radial velocities.

Traveling to the edge of the dataset and looking back at the galaxy is jaw dropping when you realize the nebulous galactic cloud is being rendered from real and observed points in Gaia Sky's database.

Closer to home, you can choose any of the planets to visit and even view the outlook from their surfaces, compete with tessellated height maps. You can speed up or slow down time through the epochs and even view everything on a variety of 3D hardware. Similarly, there's planetarium and 360-degree projection modes, either of which could put Gaia at the center of an educational installation or temporary planetarium if you have the projection hardware. You could then use the integrated Python interpreter to script your own tours through the universe. It looks incredible (and obviously needs some decent 3D acceleration to get the most out of a large dataset), and the results are clear and predictable. This is more than can be said for the view from the average light-polluted and rain-splattered English window.

### Project Website

<https://zah.uni-heidelberg.de/institutes/ari/gaia/outreach/gaiasky/>



**1 Camera:** Not just for seeing, you can also use the camera view to make a video recording. **2 Visibility:** Easily enable and disable objects within the main view. **3 Settings:** Customize the view to suit your system and viewing preferences. **4 Datasets:** Gaia supports vast catalogs of star and object data. **5 Objects:** Easily find planets and other well-known objects. **6 3D view:** See objects exactly as they might appear in space. **7 Go anywhere:** Select any star in the sky and travel to its location. **8 Properties:** Thanks to ESA's Gaia sky survey, every object comes with a wealth of data to study.

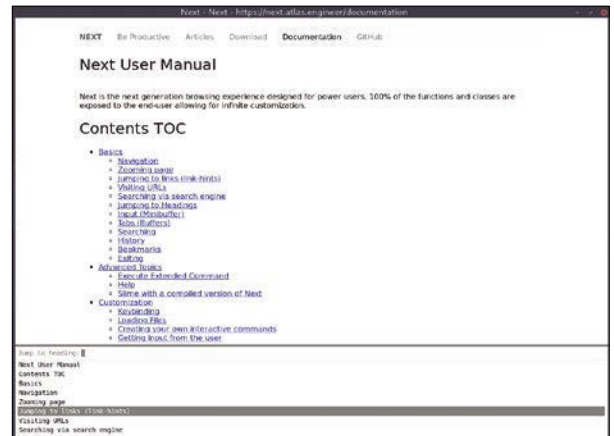
## Web browser

## Next

It's no secret that I love Qutebrowser – a Chromium/Qt-WebEngine-based web browser where you can use Vim keyboard shortcuts to do almost everything. Switching from a traditional point-and-click browser to Qutebrowser is transformative, but it's always good to see other browsers appearing with their own approaches to the same concept. The Next web browser does this, only with an even geekier bent. It's a keyboard-oriented browser with either Emacs or Vim keybindings that can be changed, drumroll please, via a configuration file and its built-in Lisp interpreter. Helpfully, when you first launch the browser, it shows you the most common shortcuts you can use to navigate the web, which is helpful, but you're on

your own after you press Ctrl+I to load a URL into the main view – although there is a lovely fuzzy search built-in when you don't know the exact URL of the page you want to load.

Speed is another big feature in Next, not just because you need to use the keyboard for everything, but because there's very little added to the raw QtWebEngine viewer that's driving the experience. Tabbed views are supported, and you can quickly switch between them with another shortcut. It's very much like working with Qutebrowser, only with different default keybindings. Another similarity is the link highlighting used to navigate to a page linked to from another page. By default, this feature hides behind Ctrl+g, and there's an excellent *Jump to head-*



The learning curve on Next is eased considerably by the excellent documentation.

ings shortcut (Ctrl+,) which opens all the headings on a page as a tab-completable list from which you can easily select. Using both means you can quickly navigate through your favorite pages, many of which work just as well as with something like Chromium. Unlike Qutebrowser, though, there's no command or visual edit mode, but that makes this more of a pure browser than a "browser with aspects of an editor," which should help Next win a whole alternative set of fans.

## Project Website

<https://github.com/atlas-engineer/next>

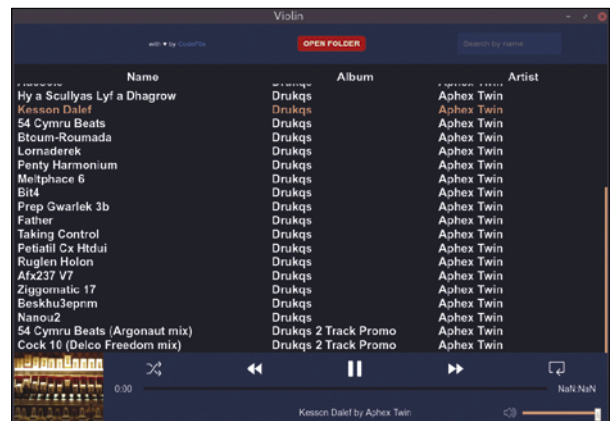
## Minimalist music player

## Violin

Minimalism isn't just useful when exploring new perspectives on web browsers, like the Next browser, or on the command line when playing with email clients. It's a great way of reassessing how we all use our favorite applications and application types. Music players are a great example, because the task they perform – playing music – is completely independent of how you might interact with the player. There's a good argument for a player simply playing music in the background while keeping user interaction to an absolute minimum. Even most of the features in Clementine, which was itself a project initiated to cut many of the superfluous features in Amarok, probably aren't needed by

most users. This is why Violin is such an interesting new project. It's a music player that's so minimal, it doesn't even include a single configuration option.

When you first load Violin, you're presented with a simple GTK3+-based user interface (UI) asking you to choose a folder to locate the music you want to play. Violin will play MP3, FLAC, MP4, WAV, and OGG files via PulseAudio. Having added the files, the music queue is simply populated with its contents, with any file-embedded album artwork used as a thumbnail. You just need to click *Play* to start playback, or use the media keys on your keyboard even when the application is running in the background. There's no playlist support, no 3D visualizations, no ability to create a library of your fa-



There isn't a single hidden feature in Violin. What you see, or hear, is exactly what you get.

avorite tracks, and no online search for more metadata. Depending on how you listen to music, these may take minimalism a step too far. But if you're used to organizing your music into one folder per album, Violin's approach is perfect. It's also cross-platform, which means you get exactly the same fast experience on Windows, Linux, and macOS.

## Project Website

<https://violin-player.cc/>



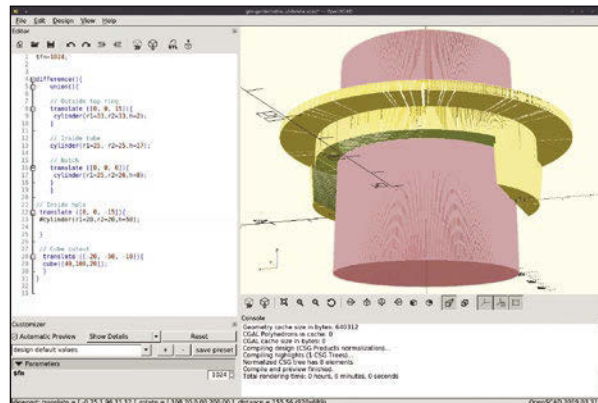
Computer-aided design

# OpenSCAD

When most of us think of a CAD application for designing real physical things in two and three dimensions, we think of tools that are a little like Blender but tuned for mechanical engineering. They might include dragging and dropping objects with mathematical names, such as a torus, cylinder, sphere, and cube, dropped onto a rigidly defined grid where you can then play with their layout using properties to create a more complex object. This is the way FreeCAD works, for example, but it's not the way OpenSCAD works. OpenSCAD does the same thing, only the drag-and-drop part is replaced by a simple scripting language. This sounds intimidating at first, as if OpenSCAD could be the design equivalent of Mathe-

matica, but it's not. The language is simple, easy to learn, and powerful enough to let you easily create complex objects. It's actually better than drag and drop in many ways.

When adding each object with code, such as `cylinder(r1=33,r2=33,h=2)`; to create a cylinder with a radius of 33 and a height of 2, you immediately see what it looks like in the 3D view. You can then group together these functions to create a combined union structure that you could then perhaps cut from a "difference" structure. When you have objects in these structures, it becomes very easy to move them with a single translate function, or copy and paste them, in ways that just aren't possible with simple point and click. This extends to



Even though the units are never specified, most 3D printers and other software take them to mean 1=1mm.

making small changes when something doesn't fit, such as increasing a radius by a single value, or moving an object to the side, which is often needed when creating models you're going to print. The next step is simply saving the model and either loading it into Blender to make more bespoke edits, or slicing the model to send to your printer.

**Project Website**  
<https://www.openscad.org/>

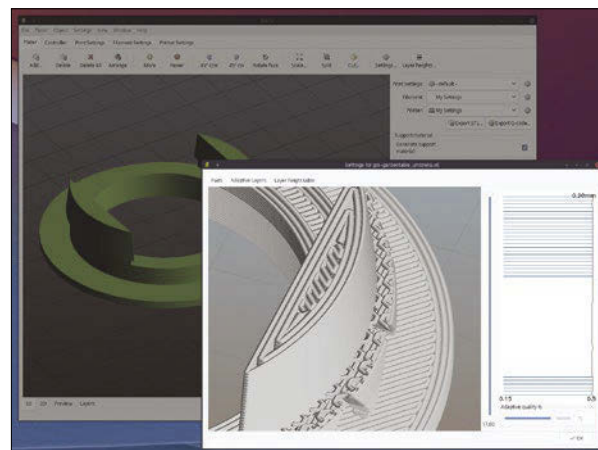
3D printing tool

# Slic3r

Staying within the realm of 3D printing, there's a process between designing your model and printing that's called "slicing." This simple word hides a hugely complex idea: Slicing takes your 3D models and turns them into simple instructions for a 3D printer to follow. Those instructions are typically formatted as G-code, and they consist of thousands of lines like `G1 X10 Y20 F1200` to tell the printer to go to position 10,20 at a speed of 1200. It's like you're controlling a huge three-dimensional turtle. To do this, a slicer needs to know about your printer and your filament before it can deconstruct your model into the commands required to reconstruct. It also needs to know

how your printer handles the infill and the tricky issue of adhesion to the bottom layer, and whether the bridged parts of your model will need artificially constructed supports. Slic3r can do all this and more.

Slic3r starts off with a configuration assistant, which lets you tell it all about your printer and the temperatures you want for the filament and the heating bed. You can now import your 3D model, usually as an STL file. There are simple operations to enable you to rotate the model for best alignment with the platter, and then you decide whether you need a brim, skirt, or support material. There's a variety of values for the infill, including 2D and 3D algorithms, before you set a percentage density for your



We've barely covered the first layer of Slic3r's capabilities, including toolpath previews, its command-line interface, and multi-extruder (color) support.

model. More dense equals stronger and heavier, but also more use of the sometimes costly filament. Thanks to Slic3r, however, you can see how much each model is going to cost, and how long it'll take to print.

**Project Website**  
<https://slic3r.org/>

## Image viewer

# Geeqie

**X**v, the perfect image viewer, was created 25 years ago, and it's a tool that's still being developed and that you can still install on your Linux box. However, that doesn't stop other developers from trying new approaches, which isn't a bad thing with something as ubiquitous as image viewing. We all need something that appears quickly when we select an image, is compatible with the majority of image formats, and maybe provides a few simple processing tools in case an image needs an edit before sharing. Geeqie could be that application. It's a fork of GQview, which is no longer being developed, but is now built atop GTK+3 and is crammed with features.

Launching without an image as an argument lets you use the

UI to navigate through your file-system to your images. As soon as you open a folder containing an image, it appears in the preview pane almost instantly. If there's more than one image, a click of the mouse or a press of the space bar will load the next image. GTK gives the UI a very slick and minimal look, perfect for image viewing, without compromising on functionality. You can switch the file view to an ultra-fast thumbnail view, which works perfectly with large high-resolution images. Images can even be shown as a calendar, alongside a color histogram, and as convenience features such as under- and over-exposure highlighting. You can perform 90-degree rotations and use external plugins for essential editing,



**Geeqie supports a huge number of file formats, including even arcane RAW formats used by DSLR cameras.**

such as cropping or loading into darktable. You can tag images with keywords, add comments, and use geolocation with OpenStreetMap. It's a brilliant utility that does everything you need from an image viewer, but does so quickly and without getting in the way of the image viewing.

**Project Website**  
<https://github.com/BestImageViewer/geeqie>

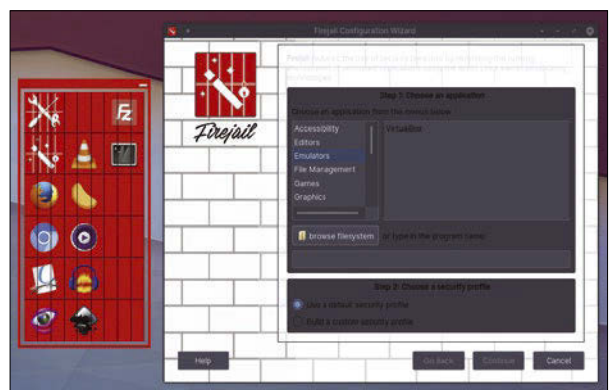
## App sandbox

# Firejail

**L**inux is secure, not just because of its relative obscurity, but because of the various levels of isolation it employs. This is something the Linux kernel is very good at, and something that's used to the fullest extent in the various virtualization and container solutions that have made Linux so popular on the cloud. However, it's not always easy to take advantage of all this isolation technology when you're a regular user. Sometimes you need ad-hoc isolation to run something of unknown provenance, and spinning up a virtual machine just to set a download is often overkill. This is where Firejail can help.

Firejail is a little like a firewall to protect your operating system from your applications,

using the power of SUID, Linux namespaces, and `seccomp` to limit an application's view of your wider system. It's like a service only having access to port 22 on a server; only with Firejail, you can isolate anything – including servers themselves. It works as a kind of sandbox and is run from the command line with the name of the command as its argument. There's a GUI too, called Firetools, which acts as a simple panel launcher, so you can quickly run your most common applications without having to enter the command line. It even includes its own process manager and configuration tool, so you can easily add applications and commands and set which security profile they're going to



**Run untrusted apps on your desktop without resorting to a virtual machine or Qemu.**

use. Another excellent feature, currently in development, is Firtunnel, which is a VPN to enable applications to communicate with one another without escaping their encapsulated environments. All of this makes Firejail a great solution for those times you need to run untrusted software on a machine you don't want sullied with binaries of unknown origin.

**Project Website**  
<https://firejail.wordpress.com/>

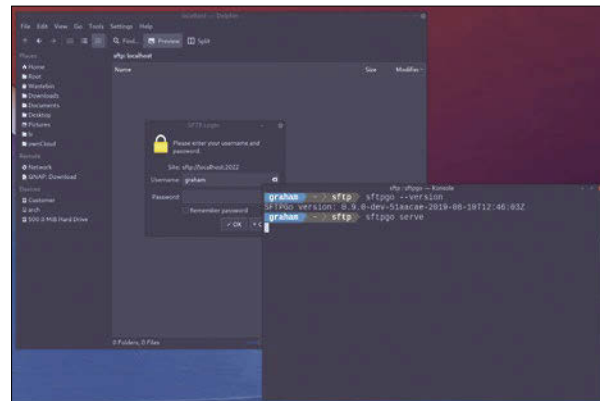
## File server

## SFTPGo

So many of us use SSH to connect to our remote machines and our multifarious embedded devices around the home, that SFTP is the easiest choice when you need to transfer files between them. It's as secure as SSH, uses SSH, and unlike `scp`, it's able to resume interrupted file transfers and list remote files much like a remote filesystem. However, unlike its FTP namesake, SFTP is still more typically used for ad-hoc transfers rather than as a way for multiple users to access a remote filesystem. That's where SFTPGo can help. SFTPGo augments SFTP with the kind of features you used to expect from an FTP server, including user accounts locked to their home directories, virtual accounts, and quota support for limiting storage, band-

width, and per-user permissions. It's been designed to run constantly in the background, acting as a secure version of an FTP server. As it's written in Go, it's easily installed with Go's `go get` package manager. It can then be run by typing `sftpgo serve` on the command line.

As with old-school FTP, you first need to create a configuration file to set up your environment. This is where you set the port on which the server runs, how authentication is going to be handled, user configuration along with usernames and passwords, and a public key, which is mandatory. There's also an HTTP-accessible Rest API, which can be used to do things like send a notification or set a trigger condition, alongside running more typical FTP com-



To get SFTPGo running, write a simple config file containing your public SSH key, and run the provided SQL script to create the database.

mands (download, upload, delete, and rename). This all makes SFTPGo sound more complicated than it really is, because, after creating the database with the script provided and the configuration file from the documented example, you have a secure and super useful file server.

## Project Website

<https://github.com/drakkan/sftpgo>

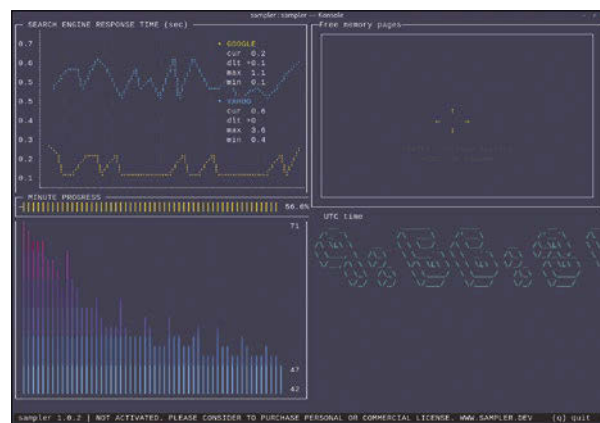
## Data visualizer

## Sampler

Sampler isn't an audio sampler for recording and playing back audio recordings, but it does take a similar approach to command-line output. Instead of sampling audio data, it's been developed to sample the output from whatever command you configure it to watch. It can then display that output using a variety of command-line visualizations, including histograms, bar charts, gauges, text boxes, and sparklines. It's configured via a simple YAML-formatted configuration file that you pass as a single argument when you run the `sampler` command. This means that you have to manually create a working configuration before you can see any of the fantastic visualizations, but

one of the best things about Sampler is that the syntax used within the file is simple to learn and powerful enough to monitor everything from simple CPU usage to remote Kubernetes container statistics.

The configuration file's syntax requires a chart type, a scale, legend definitions, and the commands to run, including the logic to parse whatever data you require. You then set a sampling rate for the frequency of the command to be run. You can add as many of these sections as you need and change the layout on the fly while Sampler is running. It all works a little like a simplified curses display engine, created specifically as a kind of command-line dashboard for people who mostly stay in the



Alongside monitoring, Sampler also allows you to trigger a script or command when a specific condition is met.

dashboard, and it works really well. The code itself is licensed under GPLv3, but the binary code that builds it includes a nag reminder in an attempt to get users to pay for a license. It's a little distracting, but if it helps with continued support for Sampler, it's worth the inconvenience.

## Project Website

<https://github.com/sqshq/sampler>



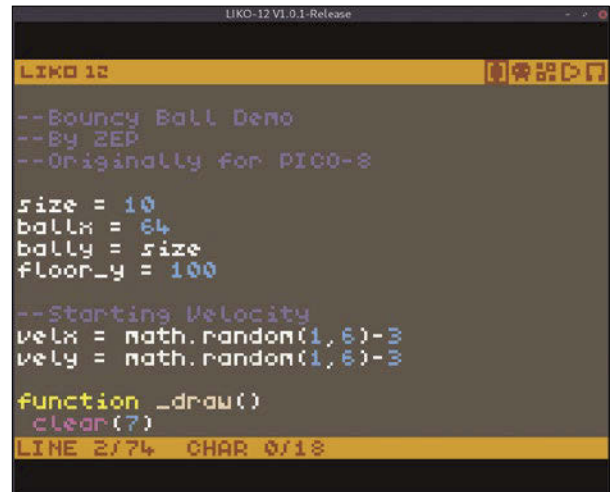
## Retro games engine

## LIKO-12

When we used 8-bit computers in the 1980s, it seemed like every upgrade to the hardware was a step away from the blocky sprites and 3-channel music that defined the era. The assumption at the time was that future technologies would hide the pixels and the number of channels, making games and computing inextinguishable from the real world around us. This has happened, to some extent. It's now almost impossible to tell if CGI is being used in films, and even games are pushing the boundaries of realism. But what has been most surprising is that the 80s-era sprites and 3-channel music have not just survived into this future, they're flourishing, and it's not because of nostalgia either. It seems that developing games

within the approximate limitations of 8-bit hardware brings out playability and design refinements that aren't possible with the infinite potential of 3D hardware.

This is why LIKO-12 is such a great idea. It works like an emulator of old 8-bit hardware, except that the hardware it's emulating is fictional. Written entirely with the Lua scripting engine, LIKO-12 itself has been built to be its own fantasy retro games platform, letting you create and play games a little like games were created and played in the 1980s – only without the 20-minute cassette loading time (or without waiting for the shared television to become available). When first launched, you're presented with a lovely pixelated text interpreter, much like an Atari 800 or Commodore 64. You can



```

LIKO-12
--Bouncy Ball Demo
--By ZEP
--Originally for PICO-8

size = 10
ballx = 64
bally = size
floor_y = 100

--Starting Velocity
velx = math.random(1,6)-3
vely = math.random(1,6)-3

function _draw()
  clear(7)
LINE 2/74 CHAR 0/18

```

With LIKO-12, you can play a lovely retro-themed game and then press *Escape* to edit its code, sprites, and levels from the integrated editor.

get quickly started by typing `install_demos` and `cd Demos`. Type `dir` to see the demos that have been installed and then `load` followed by `run` to play with one. What's amazing is that after quitting, pressing *Escape* takes you to an integrated editor, from where you can edit the code, the graphics, and the maps to create your own awesome 8-bit games!

## Project Website

<https://liko-12.github.io>

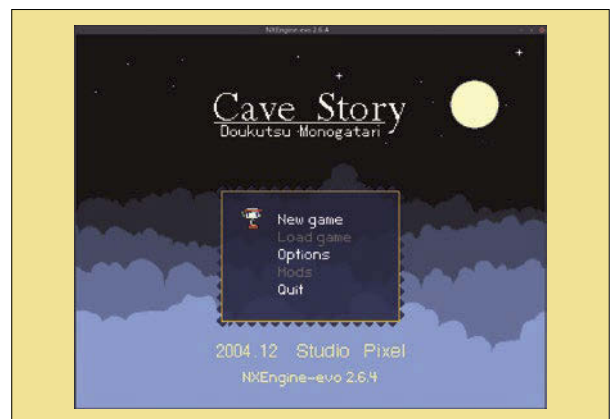
## 2D platform game

## NXEngine-evo

One of the great things about a developer going to the insane amount of trouble it takes to completely rewrite an old game for a new platform is that you don't have to worry about quality control. No one is going to go through that amount of trouble for an average game, and that's exactly the case with NXEngine-evo. It's an upgraded and refactored version of NXEngine by Caitlin Shaw, which was itself an open source clone of a 2006 game called *Doukutsu Monogatari*, or *Cave Story*. *Cave Story* itself has a fascinating origin story, being the product of a passion project by a sole indie developer over years of work. The end result changed attitudes towards indie gaming and very much paved

the way for the new pixelated platformer renaissance we're currently enjoying.

*Cave Story* is a brilliantly playable 2D platform game with beautifully hand-crafted graphics and an evolving plot that starts with the central protagonist waking in a cave with no memory of previous events. There are problems to be solved and weapons to be collected as the player pieces everything together and progresses through the game. It's unlikely that *Cave Story* would be playable on modern hardware and certainly not on the many platforms on which you can build NXEngine without this open source recreation. The new version is a brilliant way to experience the game – even while the original assets are still proprietary. The new en-



NXEngine-evo keeps *Cave Story* alive and working on modern hardware. If you want to help keep it going, take a look at the project's [Patreon page](#).

gine can be scaled to run on modern resolutions (up to 1920x1080) and adds support for SDL2, animated character portraits, localization, additional platform ports, and many other bug fixes. It's a big improvement over the rather outdated engine included in Libretro, and a fantastic way to play the game. It's *Cave Story*'s best incarnation yet, and the best way to experience the game if you've yet to play it through.

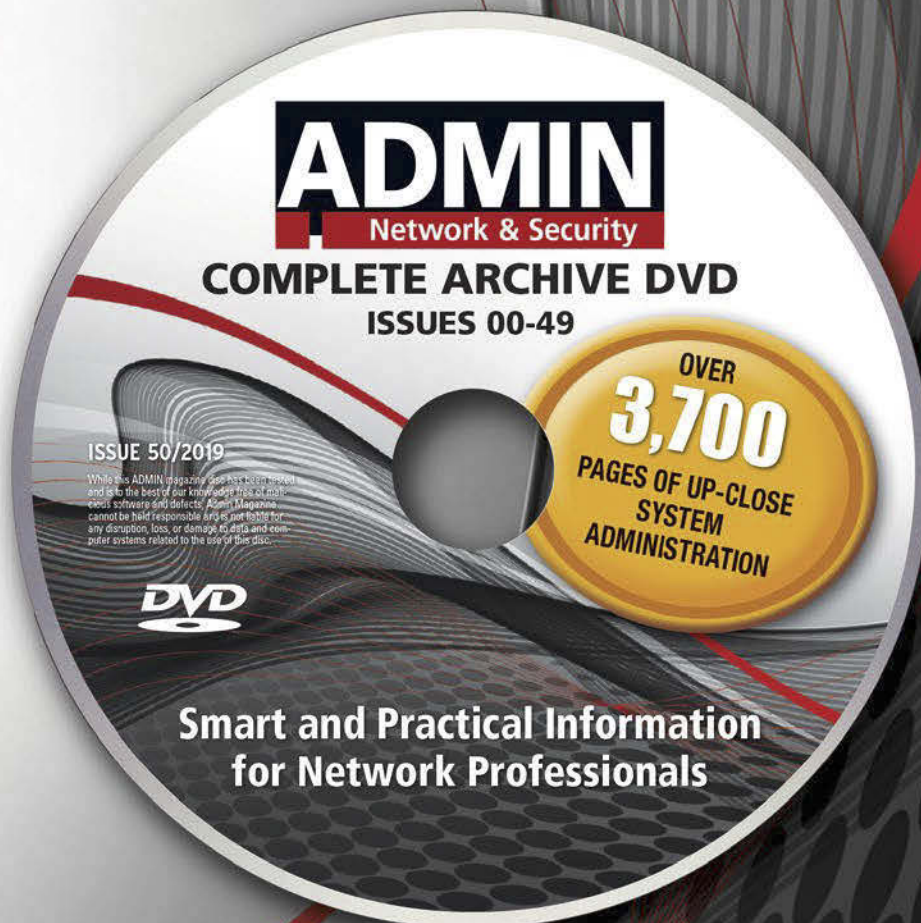
## Project Website

<https://github.com/nxengine/nxengine-evo>

# 9 Years of ADMIN on One DVD







This searchable DVD gives you 50 issues of ADMIN, the #1 source for:

- network security
- system management
- troubleshooting
- performance tuning
- virtualization
- cloud computing

Clear off your bookshelf and complete your ADMIN library with this powerful DVD!

**ORDER NOW!**  
[shop.linuxnewmedia.com](http://shop.linuxnewmedia.com)



# Mastodon clients

## Post Line

The open and simple Mastodon API makes it easy to create applications to interact with this federated microblogging platform. Here are some of the clients that the community has come up with and how you can use them.

BY PAUL BROWN

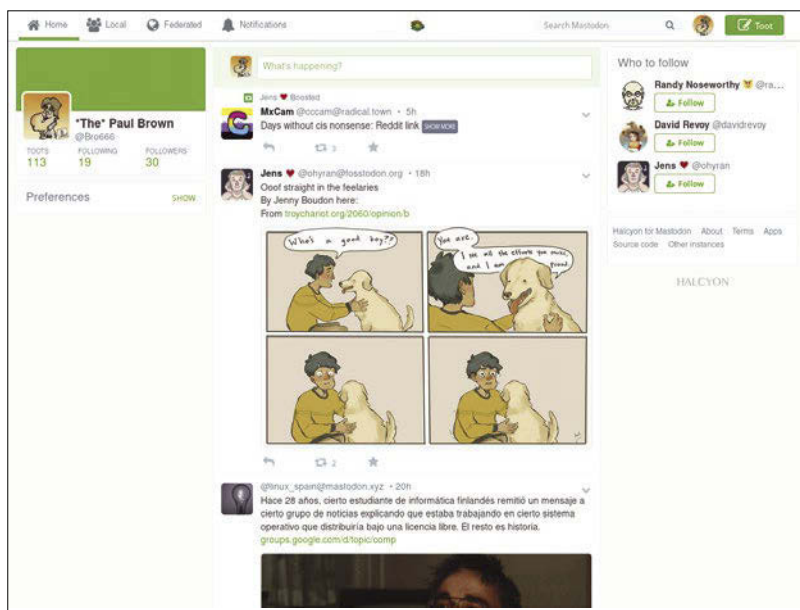
Last issue [1], I introduced the Fediverse [2] and, in particular, what is becoming the rising star in the federated universe: Mastodon [3]. Mastodon, a microblogging platform à la Twitter, is succeeding because it is simple to use, while at the same time packed with features. It comes with none of the shady shenanigans proprietary platforms inflict upon their users, and it looks slick, cool, and attractive.

Thanks to Mastodon's open and simple API, you can easily write scripts to interact with your account. This has led to a whole gaggle of web clients, desktop and mobile apps, and command-line tools for Mastodon.

### Web Clients

To begin with, there is Mastodon's own default web interface. Your first interaction with Mastodon is probably going to be through this. It comes with a lot of features, so many that I dedicated a large chunk of last month's article to explaining all of Mastodon's ins and outs. That said, there are more web clients out there.

**Figure 1:** Halcyon is a web client that makes your Mastodon feed look like the old Twitter.



Halcyon [4] is one of them. Halcyon is good if you want to enjoy the simplicity of the good, old Twitter look and feel (Figure 1) – from back when Twitter actually looked good.

Then there is Mastodon Scheduler [5]. More than a web client for Mastodon, Mastodon Scheduler is a web application, one that lets you schedule toots to be posted at a later date. A word of warning, you cannot add attachments to posts submitted with Mastodon Scheduler.

For Mastodon Scheduler to work, you need to register a new application with your account. Log in to your Mastodon account, go to *Settings | Development* and click the *New application* button. Fill in the *Application name* text box with, say, *Scheduler* (although the name does not seem to matter all that much), and click *SUBMIT* at the bottom of the page. This will log your application with the federated network.

Now, if you click on your application's name, it will show a list of three keys at the top of the page. The key labeled *Your access token* is what you need to input in the *Access token* text box on the Mastodon Scheduler form.

Finally, Pinafore [6] is another web-based front end, which may not seem to add anything new until you try it on a mobile device (Figure 2). Then it makes sense: On a tall, narrow screen, it makes Mastodon much less intimidating than using the default web interface.

### Mobile Apps

Then again, there are a lot of mobile apps that you can use to toot on Mastodon. Tusky [7], for example, is a good-looking, entry-level app. It allows you to manage several accounts simultaneously and post, boost, and draft toots. When you are browsing your timelines, you can swipe right or left and see another timeline, so you can swipe from your home, to notifications, to local, to the federated timeline, and then back again.

If you want something with more features, check out SubwayTooter [8]. Apart from allowing

you to manage several accounts like Tusky, the array of options is impressive – tap the hamburger menu icon in the lower left-hand corner of the app to see them all. In addition to being able to swipe through multiple columns like in Tusky, you can display two columns at the same time when you hold your phone horizontally (Figure 3) and add new columns to the list.

To do this, say you want to add a column that shows all posts that contain the `#linux` hashtag. Tap the hamburger menu (bottom left), tap the search icon (a magnifying glass), and type in your search term. That in itself already creates a new column with all the search results. However, if you tap the first result containing the hashtagged term, it will create a new column with all the toots containing the hashtag.

At the bottom of the screen, you have a kind of map that shows you where you are in the list of columns and what columns you have on the right and left. Your home timeline is represented by a house icon, notifications are a speech bubble with an exclamation mark, the local timeline is an icon

of a running man for some reason, and the federated timeline is a man on a bike.

You also have a column for liked toots (represented by a star icon) and a column for conversations with other users (two overlapping speech bubbles). A hashtag column is represented by a `#`, and a search is represented by a magnifying glass.

If you want to get rid of a column, just tap the X in the upper right-hand corner. You can reorder columns by tapping on the hamburger menu and selecting *Columns list*.

SubwayTooter’s other interesting feature is the possibility to schedule your toots directly from within the app. This is something that is built into Mastodon’s protocol (supposedly). Some servers do not support scheduled toots, and others seem a bit flaky. Either way, it’s a great idea that I am looking forward to seeing implemented more universally in the Fediverse.

Precisely to avoid the kinks, Fedilab [9], an app I started talking about last issue [1], adds one more feature: scheduling from the device itself. When you write your toot, you can choose to schedule and then decide whether you’d prefer to schedule it on the server (which, as mentioned, may not work) or let the app take charge. In my experience, the latter is more reliable. Fedilab also allows you to swipe through columns and add new ones, just like SubwayTooter.

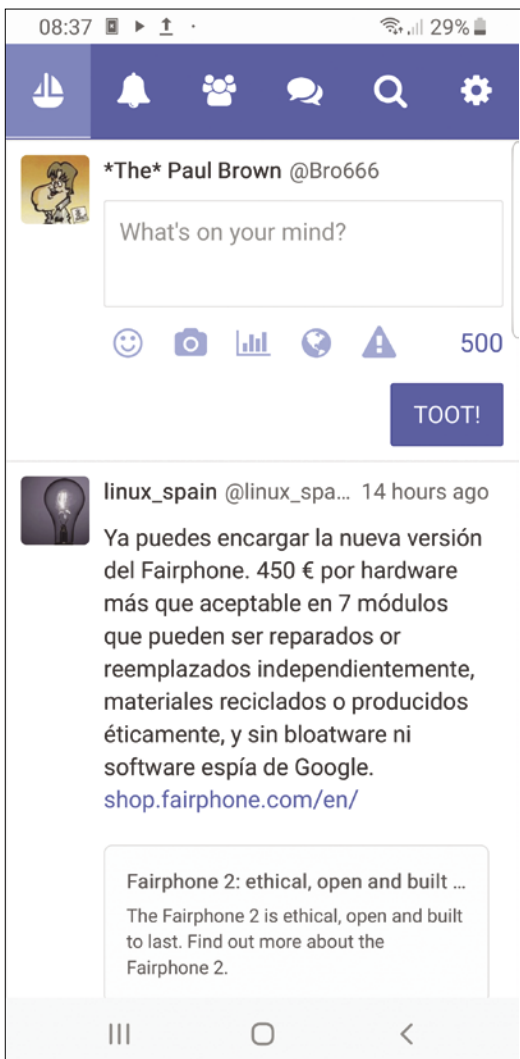
### Desktop Applications

While there are plenty Mastodon clients for mobile phones, desktop clients are pretty thin on the ground. The only one that seems to work is Whalebird [10], an Electron-based application. In other words, it is basically a thinly veiled web app wrapped in a desktop interface (Figure 4).

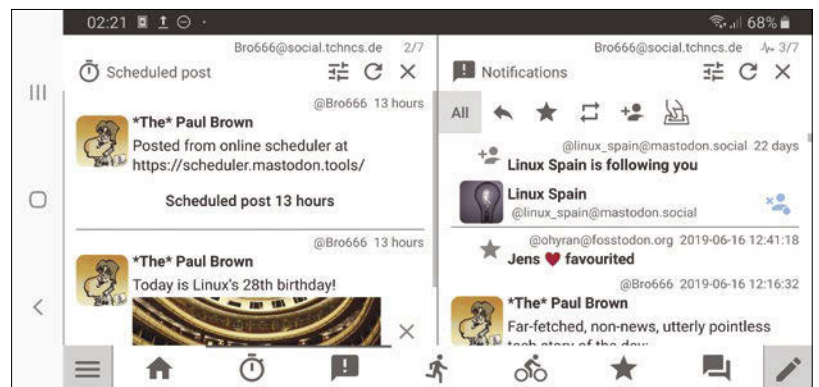
The first time you run Whalebird, you give it your instance and your login details, and it will open your Mastodon instance in a web browser and ask you to authorize the application. Once you do that, it will give you an authorization code that you copy and paste into Whalebird.

Whalebird is pretty basic. Although it supports managing several accounts, it can only show one

**Figure 3:** SubwayTooter is the only mobile Mastodon client that lets you see two timelines simultaneously.



**Figure 2:** Pinafore makes sense as a web interface for mobile devices.



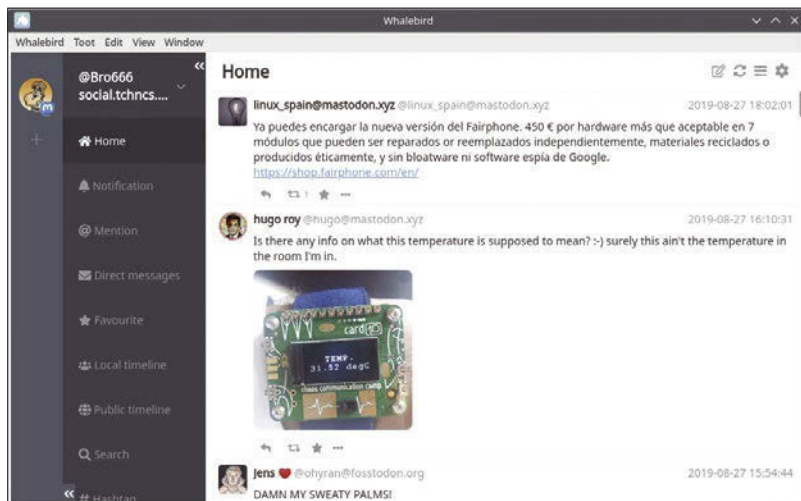


Figure 4: Whalebird is a desktop Mastodon application built with Electron.

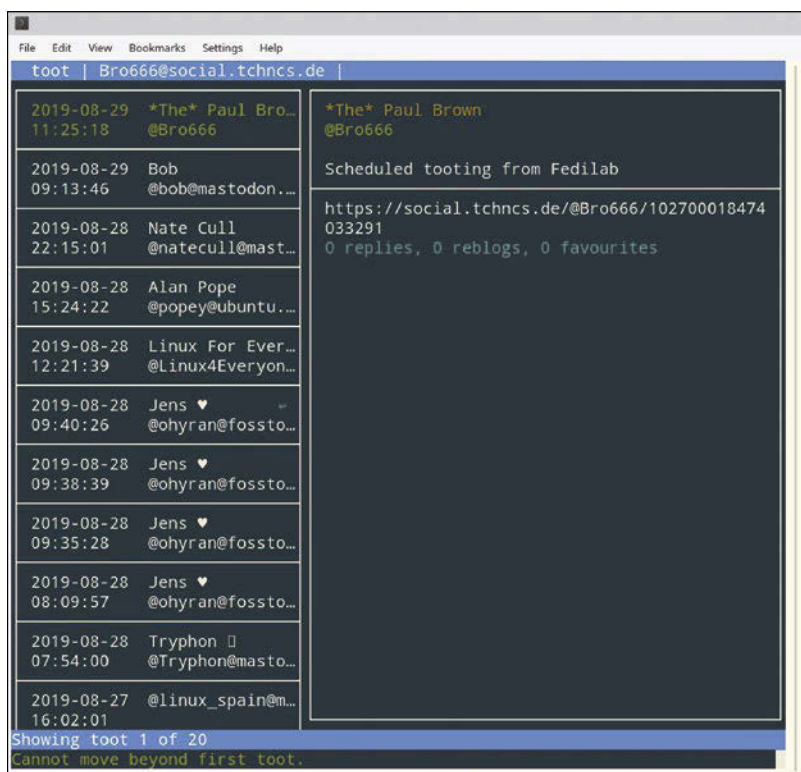
column at a time, and there are no scheduling features. It is quite pretty to look at, though.

The other tested clients did not run, did not compile, or did not contain the boasted hooks to Mastodon.

### Command-Line Tooting

There are some interesting command-line clients. Mastotool [11], for example, allows you to log in to your account, search for keywords in your toots, and grab some stats (number of toots, followers, most boosted toot, highest scoring toot, accounts from other people with whom you frequently interact, etc.). That's it, and that is the author's stated purpose for it. It is kind of useful. However, since collecting data is its main point, the fact that the stats are not very comprehensive limits mastotool's usefulness.

Figure 5: Toot comes with an experimental curses interface that allows you to browse your timeline from the command line.



On the other hand, toot [12] is more the real deal, allowing you to post, boost, delete, and do all sorts of things from the command line.

To configure your account run

```
toot login
```

and follow the instructions. As with Whalebird, at one point you will be required to authorize the toot with your Mastodon instance and copy and post an authorization code into the command line.

Once you are done, tooting with toot is pretty simple. The following

```
toot post "Tooting from toot" -m Pictures/someimage.jpg
```

will post "Tooting from toot" and the image to your timeline.

To print out all the available commands that allow you to poll your account, manage lists, follow other users, and so on, use:

```
toot help
```

Toot also has an experimental curses interface to read toots (Figure 5). You can scroll through your timeline's list, but that is about it – as far as I can see, you cannot boost, reply, or even switch to another timeline from the curses interface. Maybe these will be features added in a future version.

Also interesting is madonctl [13], especially as a tool you can integrate into your own scripts.

To log into your account, run:

```
madonctl config dump -i your_mastodon.instance -L your@email.com -P "Your secret login password" > config.yaml
```

You usually need to copy the file that this creates into `.config/madonctl/madonctrl.yaml`, although you can point madonctl to other configuration files with the `--config` option

```
madonctl account show --config .config/madonctl/myotheraccount.yaml
```

This will show stats and data from your non-default account.

If you need to change anything, you can edit the file directly. A typical madonctl configuration file looks like Listing 1.

The instance value on line 1 refers to your Mastodon instance (i.e., wherever your account is hosted – `mastodon.social`, `mastodon.technology`, etc.). You do not have to supply the `app_id` (line 2) or the `app_secret` (line 3), but you do have to get a token (line 5) from your Mastodon in-



stance. You do that by registering a new application as explained earlier when discussing Mastodon Scheduler.

Once you have completed and saved the configuration file, you can start interacting with your new account.

The command

```
madonctl account show
```

will give you data about the account you are logged into, such as the description, its ID, number of followers, number of statuses (posts), and so on. And

```
madonctl toot "Hello"
```

will post a message to your timeline.

Although neither `toot` nor `madonctl` have commands to schedule toots, it is not difficult to imagine ways of using Linux tools to do the scheduling for you. You can use the traditional `at` [14] command, for example, to do the trick (Figure 6). The disadvantage is that your machine will have to be on at the time the toot is scheduled.

## Conclusion

The variety of Mastodon clients is wide and diverse, which is a testimony to how easy it is to create applications that interact with Mastodon. Although the mobile applications seem to cover most of the features that make Mastodon great (and then some), their web, desktop, and command-line counterparts all fall short.

That's why in my next installment, I'll be showing you how to easily write your own client and give it the features you need.

Until then, toot along! ■■■

```
[paul@Rachel ]$ at NOW + 5 minutes
warning: commands will be executed using /bin/sh
at> madonctl toot "Hello"
at> <EOT>
job 2 at Thu Aug 29 22:23:00 2019
```

**Figure 6:** You can use Linux's `at` tool to schedule a future toot.

## Info

- [1] "Welcome to the Fediverse," by Paul Brown, *Linux Magazine*, issue 227, October 2019, pp. 90-94
- [2] The Fediverse: <https://fediverse.party/>
- [3] Mastodon: <https://joinmastodon.org/>
- [4] Halcyon: <https://halcyon.cybre.space/>
- [5] Mastodon Scheduler: <https://scheduler.mastodon.tools/>
- [6] Pinafore: <https://pinafore.social/>
- [7] Tusky: <https://tusky.app/>
- [8] SubwayTooter: <https://github.com/tateisu/SubwayTooter>
- [9] Fedilab: <https://fedilab.app/>
- [10] Whalebird: <https://whalebird.org/en/desktop/contents>
- [11] mastotool: <https://github.com/muesli/mastotool>
- [12] toot: <https://toot.readthedocs.io/en/latest/>
- [13] madonctl: <https://lilotux.net/~mikael/pub/madonctl/>
- [14] at: <https://www.computerhope.com/unix/uat.htm>

## Listing 1: madonctl.yaml

```
01 instance: 'https://your_mastodon.instance'
02 app_id: 'e6d56c608a350735868c7884985f1f849f24d3bf43cc316a0fdcf3b1e17bccaf'
03 app_secret: '2fd94d21455cb0f494a4ef09d994f00d84ccb025e6f576ca5f0eaaa05e5b8f9b'
04
05 token: 'f-G932jfd83erfjdfij847WIOSWksDw9sekdeII'
06 login: 'your@email.com'
07 password: 'Your secret login password'
08 safe_mode: false
09
10 #default_visibility: unlisted
11
12 #template_directory: ''
13 #default_output: theme
14 #default_theme: ansi
15 #color: auto
16 #verbose: false
17 ...
```

# FEATURED EVENTS

Users, developers, and vendors meet at Linux events around the world. We at *Linux Magazine* are proud to sponsor the Featured Events shown here. For other events near you, check our extensive events calendar online at <http://linux-magazine.com/events>.

If you know of another Linux event you would like us to add to our calendar, please send a message with all the details to [events@linux-magazine.com](mailto:events@linux-magazine.com).



## DrupalCon Amsterdam 2019

**Date:** October 28-31, 2019

**Location:** Amsterdam, Netherlands

**Website:** <https://events.drupal.org/amsterdam2019>

DrupalCon brings together thousands of people from across the globe who use, develop, design, and support the Drupal platform. Join us for this opportunity to access and engage with the decision-makers and minds who make Drupal happen.

## Linux Security Summit

**Date:** October 31 - November 1, 2019

**Location:** Lyon, France

**Website:** <https://events.linuxfoundation.org/events/linux-security-summit-europe-2019/>

The Linux Security Summit (LSS) is a technical forum for collaboration between Linux developers, researchers, and end users with the primary aim of fostering community efforts in analyzing and solving Linux security challenges.

## SC19

**Date:** November 17-22, 2019

**Location:** Denver, Colorado

**Website:** <https://sc19.supercomputing.org/>

The SC Conference brings together the international high performance computing community for an exceptional program of technical presentations, papers, informative tutorials, timely research posters, and Birds of a Feather (BoF) sessions.

## Events

JAX London 2019	October 7-10	London, United Kingdom	<a href="https://jaxlondon.com/">https://jaxlondon.com/</a>
Open Source Summit + Embedded Linux Conference Europe	October 28-30	Lyon, France	<a href="https://events.linuxfoundation.org/events/open-source-summit-europe-2019/">https://events.linuxfoundation.org/events/open-source-summit-europe-2019/</a>
ApacheCon	October 22-24	Berlin, Germany	<a href="https://aceu19.apachecon.com/">https://aceu19.apachecon.com/</a>
Sonoj Convention 2019	October 26-27	Cologne, Germany	<a href="https://www.sonoj.org/">https://www.sonoj.org/</a>
Open Source Summit + Embedded Linux Conference Europe	October 28-30	Lyon, France	<a href="https://events.linuxfoundation.org/events/open-source-summit-europe-2019/">https://events.linuxfoundation.org/events/open-source-summit-europe-2019/</a>
DrupalCon Amsterdam 2019	October 28-31	Amsterdam, Netherlands	<a href="https://events.drupal.org/amsterdam2019">https://events.drupal.org/amsterdam2019</a>
Linux Security Summit Europe	October 31-Nov 1	Lyon, France	<a href="https://events.linuxfoundation.org/events/linux-security-summit-europe-2019/">https://events.linuxfoundation.org/events/linux-security-summit-europe-2019/</a>
Open Source Monitoring Conference (OSMC)	November 4-7	Nuremberg, Germany	<a href="https://osmc.de/">https://osmc.de/</a>
Open Source Camp 4 Foreman	November 7	Nuremberg, Germany	<a href="https://opensourcecamp.de/">https://opensourcecamp.de/</a>
Linux App Summit	November 12-15	Barcelona, Spain	<a href="https://linuxappsummit.org/">https://linuxappsummit.org/</a>
Linux Presentation Day 2019	November 16	Cities across Europe	<a href="http://l-p-d.org/en:start">http://l-p-d.org/en:start</a>
SC19	November 17-22	Denver, Colorado	<a href="https://sc19.supercomputing.org/">https://sc19.supercomputing.org/</a>
KubeCon + CloudNativeCon North America 2019	November 18-21	San Diego, California	<a href="https://events.linuxfoundation.org/events/kubecon-cloudnativecon-north-america-2019/">https://events.linuxfoundation.org/events/kubecon-cloudnativecon-north-america-2019/</a>
DevOpsDays Berlin 2019	November 27-28	Berlin, German	<a href="https://devopsdays.org/events/2019-berlin/welcome/">https://devopsdays.org/events/2019-berlin/welcome/</a>

# CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to [edit@linux-magazine.com](mailto:edit@linux-magazine.com).



## Authors

Erik Bärwaldt	24
Swapnil Bhartiya	8
Paul Brown	92
Zack Brown	12
Bruce Byfield	28, 62
Joe Casad	3
Mark Crutch	71
Christopher Dock	50
Friedhelm Greis	66
Karsten Günther	80
Jon "maddog" Hall	73
Charly Kühnast	26
Christoph Langner	16
Andrew Malcolm	56
Vincent Mealing	71
Anzela Minosi	20
Graham Morrison	84
Shashwat Pant	36
Mike Schilli	42
Tim Schürmann	74
Ferdinand Thommes	46
Stefan Wintermeyer	32

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:

[http://www.linux-magazine.com/contact/write\\_for\\_us](http://www.linux-magazine.com/contact/write_for_us).

**NOW PRINTED ON** recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

## Contact Info

### Editor in Chief

Joe Casad, [jcasad@linux-magazine.com](mailto:jcasad@linux-magazine.com)

### Copy Editors

Amy Pettie, Megan Phelps

### News Editor

Swapnil Bhartiya

### Editor Emerita Nomadica

Rita L Sooby

### Managing Editor

Lori White

### Localization & Translation

Ian Travis

### Layout

Dena Friesen, Lori White

### Cover Design

Dena Friesen

### Cover Image

© Alexander Bedrin, 123RF.com

### Advertising

Brian Osborn, [bosborn@linuxnewmedia.com](mailto:bosborn@linuxnewmedia.com)  
phone +49 89 3090 5128

### Marketing Communications

Gwen Clark, [gclark@linuxnewmedia.com](mailto:gclark@linuxnewmedia.com)  
Linux New Media USA, LLC  
2721 W 6th St, Ste D  
Lawrence, KS 66049 USA

### Publisher

Brian Osborn

### Customer Service / Subscription

For USA and Canada:  
Email: [cs@linuxpromagazine.com](mailto:cs@linuxpromagazine.com)  
Phone: 1-866-247-2802  
(Toll Free from the US and Canada)

For all other countries:  
Email: [subs@linux-magazine.com](mailto:subs@linux-magazine.com)

[www.linuxpromagazine.com](http://www.linuxpromagazine.com) – North America

[www.linux-magazine.com](http://www.linux-magazine.com) – Worldwide

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the disc provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2019 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media USA, LLC, unless otherwise stated in writing.

Linux is a trademark of Linus Torvalds.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Ziebländstr. 1, 80799 Munich, Germany.



**Approximate**  
UK / Europe Nov 02  
USA / Canada Nov 29  
Australia Dec 30  
**On Sale Date**

Issue 229 / December 2019

# Graphics Tools

Beyond Gimp and Inkscape are a collection of powerful open source tools that will give your images a professional edge. Next month we explore some tools and techniques for better Linux graphics.

## Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

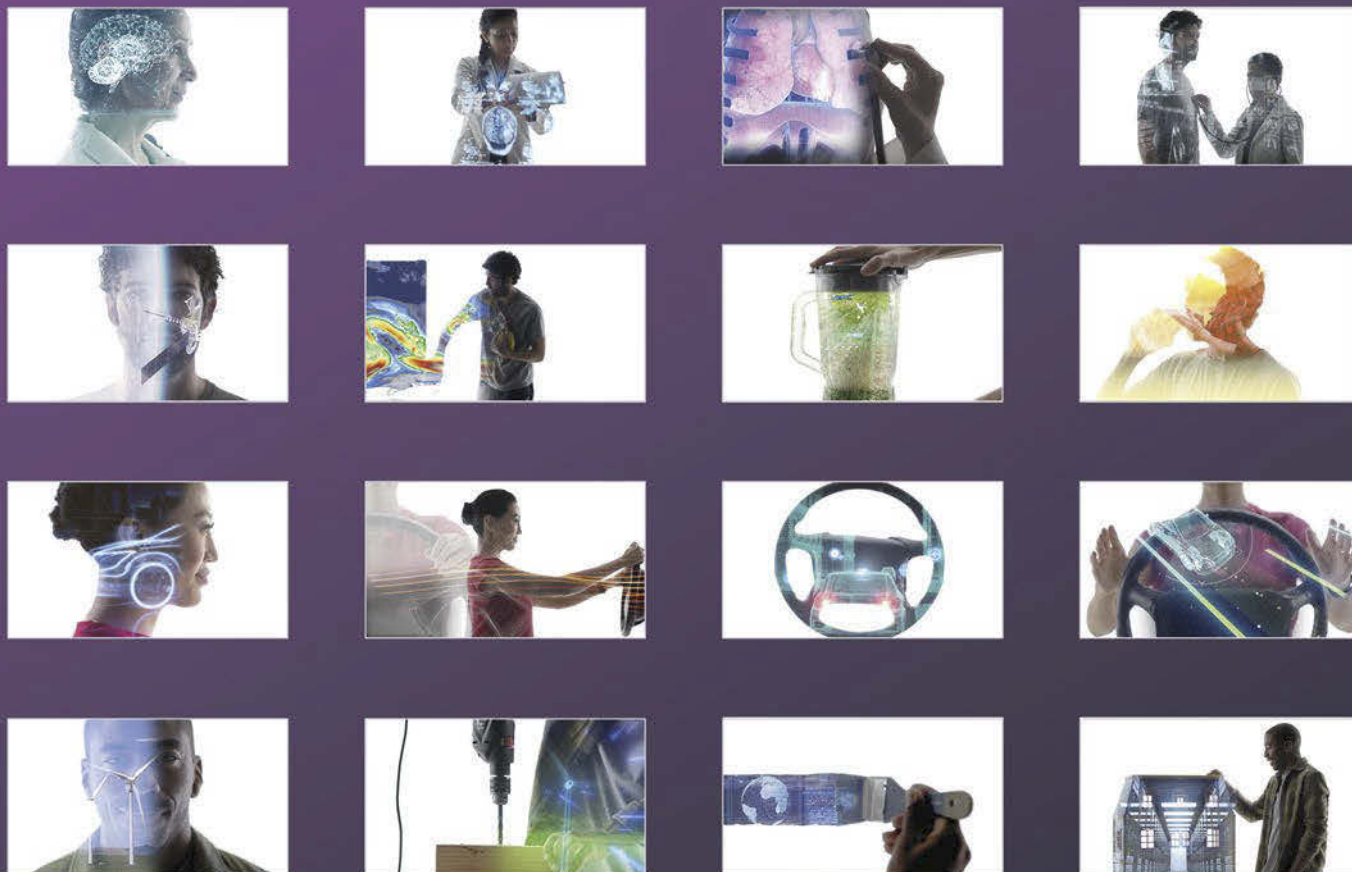
Sign up at: [www.linux-magazine.com/newsletter](http://www.linux-magazine.com/newsletter)

Image © Buchachor Pethanya, 123RF.com



# Get hyped.

Register for SC19 today and discover why HPC is now.



Calling all HPC researchers, engineers, and students – don't miss the broadest program in HPC, the massive exhibition, networking opportunities, and more. Big HPC adventures await in Denver this fall!

November 17–22, 2019

The International Conference for High Performance Computing, Networking, Storage, and Analysis



# SC19

Denver, CO | hpc is now.

# When You Think Cloud Infrastructure Think Supermicro

To Scale Your Compute and Storage  
Systems and Optimize Your  
Resource Savings Featuring  
2<sup>nd</sup> Generation Intel® Xeon®  
Scalable processors



Learn More at [www.supermicro.com](http://www.supermicro.com)

© Supermicro and Supermicro logo are trademarks of Super Micro Computer, Inc. in the U.S. and/or other countries.

